

An Imbalance-Aware Machine Learning Framework for Smishing Detection in SMS-Based Mobile Banking

Kingsley Amofo

University of Ghana

Godwin Akrong

`godwin.akrong@mail.mcgill.ca`

McGill University

Ebenezer Aquah

University of Ghana

Prince Alvin Kwabena Ansah

University of Ghana

Ebenezer Owusu

University of Ghana

Research Article

Keywords: Smishing Detection, Machine Learning, Text Classification, Class Imbalance, NLP-based Data Augmentation, Model Interpretability

Posted Date: May 5th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9604232/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

An Imbalance-Aware Machine Learning Framework for Smishing Detection in SMS-Based Mobile Banking

Kingsley Amofo¹, Godwin Banafo Akrong^{2*}, Ebenezer Aquah¹, Prince Alvin Kwabena Ansah¹, Ebenezer Owusu¹

¹School of Physical and Mathematical Sciences, Department of Computer Science
University of Ghana, Accra, Ghana,

^{2*}School of Information Studies, McGill University, Montreal, Canada

kamofo001@st.ug.edu.gh, godwin.akrong@mail.mcgill.ca*, eacquah071@st.ug.edu.gh,
pakansah@ug.edu.gh, ebeowusu@ug.edu.gh

Abstract

The increasing reliance on Short Message Service (SMS) for mobile banking and money transactions has heightened the risk of SMS phishing (smishing), posing a significant cybersecurity threat. A major challenge in smishing detection is the imbalance in SMS datasets, where malicious messages are vastly outnumbered by legitimate ones, often leading to poor detection of the minority class. This study develops and evaluates a machine learning framework for smishing detection, comparing the performance of Naive Bayes and Random Forest classifiers. To address class imbalance, the minority spam class was augmented using advanced text augmentation techniques, followed by standardized text preprocessing and feature extraction using Bag-of-Words and Term Frequency–Inverse Document Frequency (TF-IDF) methods. The models were evaluated using metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). Results show that Random Forest outperformed Naive Bayes, achieving an F1-score of 0.969, a precision of 0.995, and a recall of 0.946. The study shows that classical machine learning models, combined with text-aware data augmentation, offer an effective and interpretable solution for smishing detection in mobile banking environments.

Keywords: Smishing Detection, Machine Learning, Text Classification, Class Imbalance, NLP-based Data Augmentation, Model Interpretability.

1. Introduction

The widespread integration of mobile communication and digital financial services has revolutionized modern banking, particularly in regions where Short Message Service (SMS) remains the primary medium for transactional notifications and mobile money services. However, this reliance on SMS has also introduced significant cybersecurity risks, with SMS phishing commonly referred to as smishing, emerging as a critical threat to financial ecosystems [1]. Smishing attacks exploit users' trust in text-based communications by disseminating fraudulent messages that impersonate trusted entities such as banks, telecommunications providers, or mobile money operators [2]. These attacks aim to deceive users into divulging sensitive personal or financial information, posing severe risks to individuals and organizations alike.

Detecting smishing is inherently challenging due to the unique characteristics of SMS messages, including their brevity, informal language, and limited contextual cues [3]. These factors complicate the extraction of robust signals for classification and increase the likelihood of false negatives, which can have dire consequences in financial environments [4]. Furthermore, SMS datasets are typically characterized by significant class imbalance, with legitimate (ham) messages vastly outnumbering malicious (spam) ones [5], [6]. This imbalance often results in biased models that prioritize majority-class accuracy at the expense of minority-class detection, leaving users vulnerable to undetected smishing attacks [7]. While machine learning has demonstrated effectiveness in spam detection [8], existing studies often overlook critical challenges in smishing detection. Many approaches rely on generic oversampling techniques, which are ill-suited for textual data, or focus on deep learning models that, while powerful, lack transparency and pose challenges for interpretability in security-critical applications [9]. Classical machine learning models, despite their computational efficiency and established performance, remain underexplored in unified pipelines that integrate modern natural language processing (NLP) techniques with interpretable modeling practices.

To address these challenges, this study proposes a robust and interpretable smishing detection framework that leverages two complementary classical machine learning models: Naive Bayes [10] and Random Forest [11]. These models are integrated into a unified pipeline that incorporates linguistically informed data augmentation using the NLPAug library, standardized text preprocessing, and feature extraction through Bag-of-Words and Term Frequency–Inverse Document Frequency (TF-IDF) representations. By systematically comparing the performance of these models under identical experimental conditions, this study evaluates their effectiveness in detecting smishing messages while maintaining interpretability. The contributions of this research are threefold. First, it demonstrates the efficacy of NLP-based text augmentation in addressing class imbalance and improving minority-class detection performance. Second, it highlights the inherent interpretability of classical machine learning models, such as feature importance in Random Forest and token-level probability contributions in Naive Bayes, to provide transparent insights into classification decisions. Third, it delivers a reproducible and deployment-oriented framework that bridges modern NLP preprocessing, classical machine learning, and transparent

model interpretation, underscoring its relevance for securing SMS-based mobile banking systems in high-risk and resource-constrained environments.

The remainder of this paper is structured as follows: Section 2 reviews related work on smishing detection, machine learning-based spam filtering, class imbalance handling, and interpretability in security-oriented models. Section 3 describes the method used in the study, including the proposed approach, dataset, preprocessing pipeline, NLP-based data augmentation strategy, feature extraction methods, and model implementation. Section 4 presents experimental results, analyzing model performance and interpretability outcomes. Finally, Section 5 presents the discussion, and a conclusion is drawn in Section 6.

2. Literature review

2.1 Machine Learning Approaches to Smishing Detection

Early research on automated spam filtering mainly looked at email communications. This was because it was easy to set up and worked well with the metadata-rich structure of emails [12]. As adversaries evolved to bypass these static defenses, the field shifted towards machine learning paradigms that dynamically learn discriminative patterns from labeled data, thereby enhancing adaptability to emerging threats [13]. With the widespread use of mobile messaging, phishing threats have extended to SMS, leading to the emergence of smishing as a distinct category [14]. Smishing detection presents particular difficulties due to the brevity, informal language, and limited contextual cues in SMS messages, which constrain robust signal extraction and increase susceptibility to mimicry of legitimate transactional formats [15]. To address these challenges, contemporary studies approach smishing detection as a supervised binary text classification task, employing classical machine learning models such as XGBoost, logistic regression, support vector machines (SVM), and CNNs such as ResNet50 and VGG16 [16]. These models are typically integrated with lexical feature engineering techniques like Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), which are effective in sparse, short-text domains by efficiently capturing token frequencies and discriminative weights [17].

Naive Bayes remains favored for its computational efficiency and effectiveness on high-dimensional sparse data, operating under the independence assumption to compute probabilistic token contributions [18]. Random Forest is an ensemble method that makes models more robust by combining several decision trees to model non-linear interactions and reduce overfitting. This is based on ideas from [11] and recent uses in text security [8]. However, methodological heterogeneity persists across studies, including variations in preprocessing, feature selection, dataset splits, and evaluation, which impedes comparability and may exaggerate performance if imbalance or validation rigor is not adequately addressed [19]. In security-critical applications such as mobile banking, this underscores the necessity for standardized, operationally focused benchmarks that prioritize minority-class detection over aggregate accuracy.

2.2 Class Imbalance in SMS Spam Datasets

Class imbalance is a characteristic feature of SMS spam and smishing datasets, where legitimate (ham) messages significantly outnumber malicious (spam) ones [8]. This frequently leads to biased models that excel in predicting the majority class yet falter with significant minority cases. Such skew can lead to deceptively high accuracy while allowing high false-negative rates for smishing, thereby increasing risks in cybersecurity contexts where undetected threats can cause financial or data losses. Mitigation strategies are categorized into algorithm-level and data-level approaches [20]. Algorithm-level techniques include cost-sensitive learning, class weighting, and threshold tuning to asymmetrically penalize minority misclassifications. Data-level methods change the training distributions by either undersampling the majority or oversampling the minority. The Synthetic Minority Over-sampling Technique (SMOTE) [21] is a basic oversampling method that works well with tabular data but often makes text that doesn't make sense when used with sparse representations. It does this by interpolating synthetic samples in feature space. Consequently, recent research advocates for text-specific oversampling techniques, such as synonym substitution and random token operations, which maintain linguistic validity and enhance minority diversity [22]. Despite these advancements, many studies apply imbalance handling inconsistently or fail to conduct in-depth analyses of its effects on recall, precision-recall trade-offs, and false-positive rate metrics, which are crucial for user trust in mobile filtering [23]. Such evidence underscores the necessity for rigorous, text-aware imbalance strategies evaluated through balanced, security-oriented perspectives.

2.3 Text Augmentation and Interpretability in Smishing Detection

Text augmentation enhances generalization in NLP classification by introducing controlled variations into datasets, which is particularly advantageous for small, imbalanced, or noisy corpora such as SMS [24]. There are many different techniques, from simple lexical changes (like replacing, adding, or deleting synonyms) to more advanced contextual or generative methods. Their effectiveness depends on how well they keep the meaning [25]. Tools such as NLPAug facilitate these transformations, enabling seamless integration into pipelines for spam detection while mitigating overfitting and improving minority-class representation [26]. In smishing research, augmentation is valued for increasing spam diversity without requiring new labeling, but evaluations often extend beyond accuracy to assess impacts on precision-recall balance and operational metrics, such as false positives, which affect user experience in banking applications [27]. Interpretability is increasingly prioritized in security domains for auditing, debugging, and regulatory compliance [27], [28]. Intrinsic interpretability through feature importances in RF or token probabilities in NB delivers immediate insights without post-hoc tools, aiding analysis of decision drivers such as lexical cues. However, many studies report metrics without analyzing feature contributions or error patterns, limiting deployment readiness and robustness against

adaptive attacks [29]. Integrating augmentation with intrinsic interpretability thus emerges as a pathway to transparent, effective smishing frameworks.

2.4 Challenges

Despite advancements in SMS spam and smishing detection, significant challenges persist. Methodological inconsistencies in areas such as preprocessing, feature selection, data partitioning, and evaluation metrics hinder reproducibility and cross-study synthesis, often leading to inflated results without operational validation. While the issue of class imbalance is recognized, there is a lack of tailored, text-centric applications; generic methods like SMOTE continue to be used despite their limitations, and evaluations seldom thoroughly examine minority recall or error behaviors. Deep learning models, although powerful, present deployment challenges such as high resource demands and a lack of transparency, which often overshadow efficient and interpreted classical alternatives [30]. Interpretability is frequently considered secondary, with limited focus on cue analysis for error debugging or enhancing attack resilience. These gaps point to the need for unified frameworks that emphasize rigor, text-aware imbalance handling, and intrinsic interpretability to develop deployable smishing solutions in mobile finance.

3. Methods

3.1 Proposed approach

The primary objective of this study is to develop a robust and scalable machine learning–based framework for detecting smishing, also known as SMS phishing, within the context of mobile banking communications. Smishing messages exploit the ubiquity and trust associated with SMS messaging to deceive users into divulging sensitive personal or financial information, posing a severe threat to banking security infrastructures. Traditional rule-based detection systems are inadequate in mitigating these risks because of their inability to capture the linguistic diversity, adaptive behavior, and contextual variation of malicious text messages. In response, this research adopts a data-driven methodology that integrates advanced text augmentation strategies and supervised machine learning models to achieve a high degree of classification accuracy and generalizability. The focus is on leveraging the complementary strengths of the Random Forest (RF) and Naive Bayes (NB) classifiers, enabling both interpretability and robust handling of high-dimensional text data.

The methodology begins with the formalization of smishing detection as a supervised binary classification problem. Let $D = \{(x_i, y_i)\}_{i=1}^m$ represent a labeled dataset of SMS messages, where each $x_i \in \mathbb{R}^n$ is a feature vector corresponding to an SMS message and $y_i \in \{0,1\}$ denotes the binary class label, with 0 representing legitimate messages (ham) and 1 representing malicious smishing messages (spam). The task is to approximate a classification function $f: x \rightarrow y$, where $X \subseteq \mathbb{R}^n$ and $Y = \{0,1\}$, such that for any unseen input x_i , the prediction $\hat{y}_i = f(x_i)$ minimizes the classification error. The optimization objective is formalized as minimizing the empirical risk:

$$R(f) = \frac{1}{m} \sum_{i=1}^m \ell(f(x_i), y_i), \quad (1)$$

where $\ell(\cdot)$ denotes the loss function, chosen as the binary cross-entropy loss due to its suitability for probabilistic classifiers:

$$\ell(y, \hat{y}) = -(y \log y + (1 - \hat{y}) \log(1 - \hat{y})). \quad (2)$$

The classification pipeline thus aims to identify the hypothesis $\hat{h} \in H$, from a hypothesis space H , that minimizes this expected risk:

$$\hat{h} = \arg \min_{h \in H} E(x, y) \sim P_{data}[\ell(h(x), y)], \quad (3)$$

ensuring that the learned model generalizes well to unseen samples drawn from the true data distribution P_{data} .

A unique challenge in this domain is the highly imbalanced nature of the dataset, where legitimate messages far outnumber malicious ones. In this study, 86.59% of the samples were ham messages, while spam messages represented only 13.40%, creating a significant skew that could bias classification models toward predicting the majority class. To address this issue, extensive text data augmentation was applied to the minority spam class using the NLPAug Python library. Augmentation techniques, including synonym replacement, contextual word substitution, and random insertion, were employed to synthetically expand the number of spam samples. If m_{spam} denotes the original number of spam samples and m_{aug} represents the artificially generated spam samples, the augmented dataset size can be expressed as:

$$m_{new} = m_{ham} + (m_{spam} + m_{aug}), \quad (4)$$

where m_{ham} is the number of legitimate messages. This augmentation significantly improved the balance of class distributions, reducing the bias in model training and allowing both Random Forest and Naive Bayes classifiers to learn more representative decision boundaries.

In selecting classifiers, Naive Bayes was chosen for its computational efficiency and strong performance on high-dimensional, sparse data, such as text represented in a bag-of-words or term-frequency inverse document frequency (TF-IDF) format. The classifier is grounded in Bayes' theorem:

$$P(y | x) = \frac{P(x | y)P(y)}{P(x)}, \quad (5)$$

where $P(y)$ is the prior probability of the class label y , $P(x | y)$ is the likelihood of observing feature vector x given class y , and $P(x)$ is the evidence term. The “naive” independence assumption simplifies the likelihood computation to:

$$P(x | y) = \prod_{j=1}^n P(x_j | y), \quad (6)$$

where x_j represents the j – *th* feature or token, and n is the vocabulary size. Despite this strong independence assumption, Naive Bayes has been demonstrated to perform remarkably well in text classification due to its ability to handle sparsity and large feature spaces effectively.

Complementing Naive Bayes, Random Forest was employed as a robust ensemble method capable of modeling complex, non-linear relationships between features. Random Forest constructs a set of decision trees $\{T_b\}_{b=1}^B$ trained on bootstrap samples, and predictions are obtained via majority voting across the ensemble:

$$\hat{y} = \text{mode}\{T_b(x)\}_{b=1}^B. \quad (7)$$

Each decision tree partitions the feature space by recursively selecting splits that maximize the reduction in node impurity, quantified using the Gini index:

$$G(t) = 1 - \sum_{k=1}^K p_{k,t}^2, \quad (8)$$

where $p_{k,t}$ is the proportion of samples belonging to class k at node t , and $K = 2$ in this binary classification setting. The ensemble approach effectively reduces overfitting while improving classification accuracy, especially on imbalanced and noisy datasets.

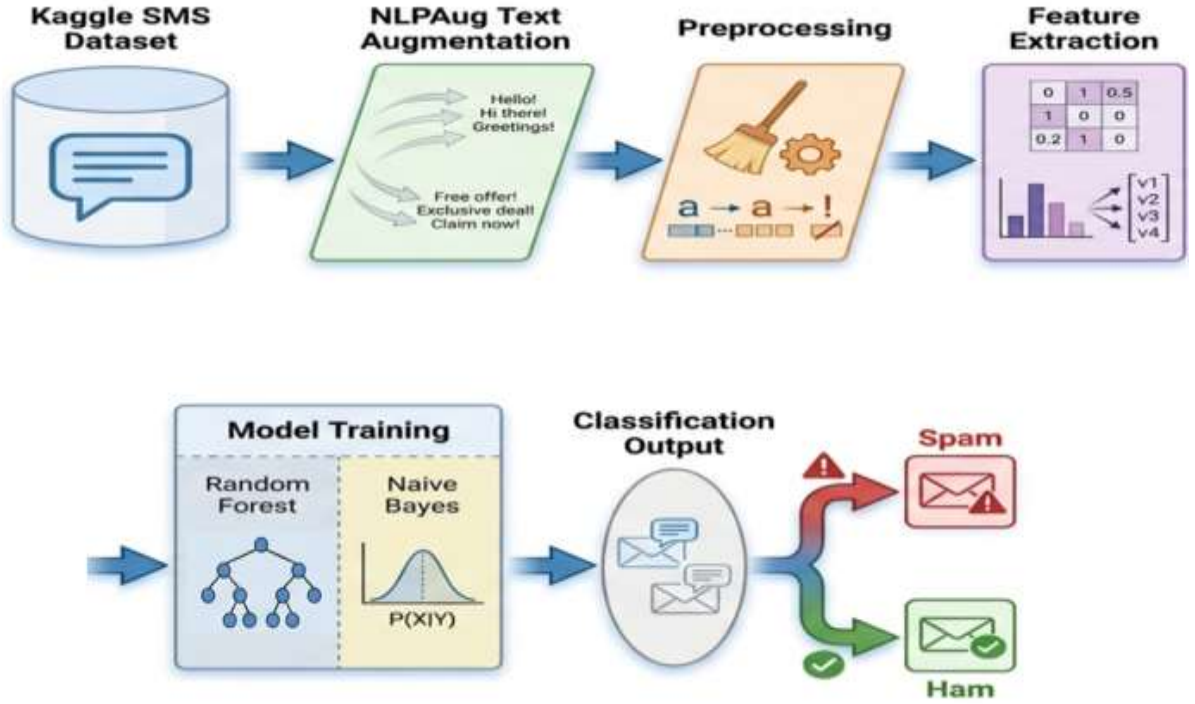


Fig. 1. Architecture of the proposed mobile-banking smishing-detection pipeline—Kaggle SMS dataset → NLPAug text augmentation → preprocessing → feature extraction → model training (Random Forest, Naive Bayes) → classification output (Ham/Spam).

The proposed pipeline thus integrates data augmentation, feature engineering, and supervised learning in a cohesive framework. SMS messages are first transformed into feature vectors through tokenization and vectorization techniques. The augmented and pre-processed dataset is then used to train both Naive Bayes and Random Forest models independently, allowing for a comparative performance analysis. The entire workflow can be conceptualized as an iterative optimization process, where the classifier parameters are updated to minimize the expected loss across both classes, ensuring that the decision function learned by the models achieves maximum discriminatory power between ham and spam messages.

This methodological framework represents a principled and scalable approach to smishing detection in mobile banking. By combining a probabilistic model with a powerful ensemble learner, the study achieves a balance between interpretability and predictive performance. The inclusion of augmentation strategies and careful handling of class imbalance further ensures that the model is robust in detecting minority-class smishing messages, which are often the most critical to identify in real-world scenarios. Such an approach demonstrates the potential of machine learning to address pressing cybersecurity challenges in the mobile financial services sector, where the stakes of false negatives are particularly high.

3.2 Dataset description

The dataset used in this study was the SMS dataset. It comprises a diverse collection of SMS messages labeled according to their nature, making it particularly suitable for research on spam and smishing detection. The dataset is structured into two primary columns: the first column, denoted as *v1*, contains categorical labels identifying each message as either *ham* (legitimate) or *spam* (unsolicited or malicious), while the second column, *v2*, holds the corresponding SMS content in raw text form. This simple yet effective structure facilitates straightforward preprocessing and feature engineering, providing a clear mapping between text data and classification targets. The dataset is composed of a total of 5,572 text messages, with 4,824 labeled as *ham* and 748 labeled as *spam*, corresponding to approximately 86.59% and 13.40% of the total samples, respectively. This pronounced imbalance mirrors real-world SMS distributions, where malicious content typically constitutes only a small fraction of overall communications, thereby underscoring the practical relevance and challenge of this research.

```
Dataset after renaming columns:
  label                                message
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                                Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

Fig. 2. Sample rows from the Kaggle SMS dataset after renaming columns; label (*ham/spam*) and message (*SMS text*).

Given its linguistic diversity and class distribution, this dataset serves as a representative benchmark for developing robust spam and smishing detection models. The raw text messages encompass a range of communication styles, from informal personal exchanges to structured marketing and phishing attempts, providing rich contextual variation for feature extraction and model training. Formally, the dataset can be expressed as a set $D = \{(x_i, y_i)\}_{i=1}^m$, where $x_i \in X$ denotes the text message content, and $y_i \in \{0,1\}$ represents the binary label, with 0 assigned to *ham* messages and 1 assigned to *spam*. The dataset's balanced representation of message content, coupled with its inherent skew in class proportions, creates an ideal experimental setting for exploring data augmentation, text representation, and classification algorithms aimed at detecting smishing in the domain of mobile banking security.

3.3 Data preprocessing

Effective data preprocessing is a critical component of this study, serving as the foundation upon which machine learning models are trained and evaluated. The raw SMS messages, as provided in the dataset, consisted of unstructured text that required systematic cleaning, transformation, and feature representation before classification. To maximize the performance of the machine learning algorithms, several preprocessing steps were applied, beginning with text augmentation to address class imbalance, followed by text normalization, feature extraction, and label encoding. These transformations were carefully designed to retain semantic meaning while mitigating noise, ultimately yielding a robust numerical representation of the input data suitable for downstream machine learning tasks.

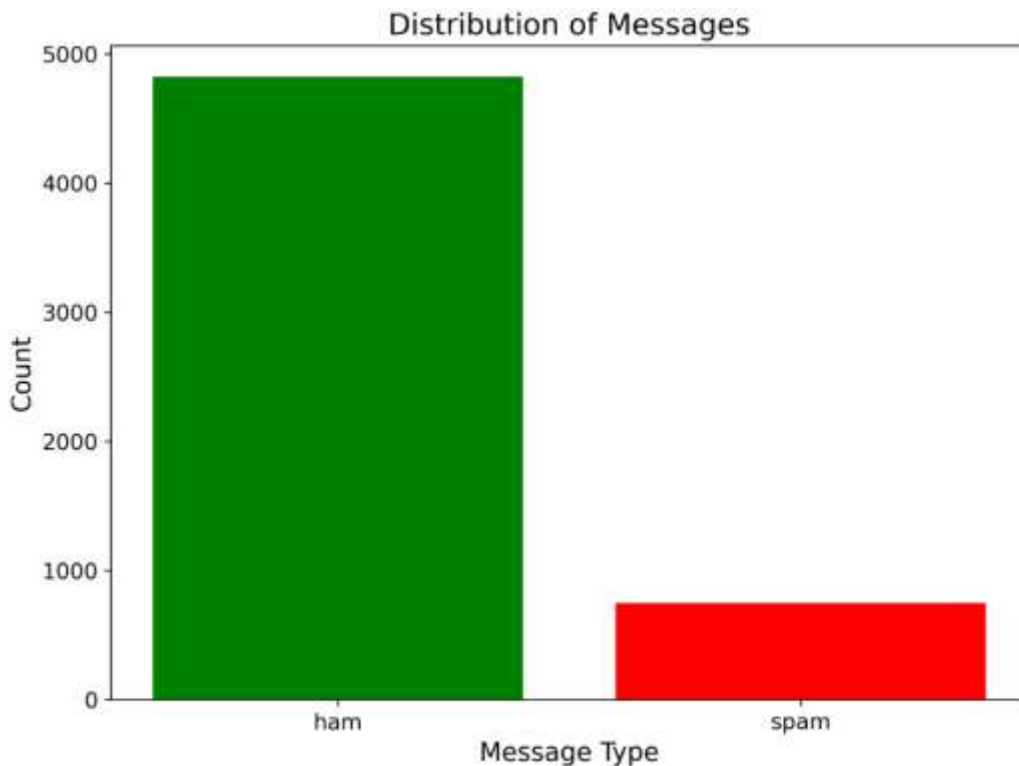


Fig. 3. Distribution of ham against spam messages.

One of the most significant challenges in this research was the severe imbalance between *ham* and *spam* messages, which could bias the classifier toward predicting the majority class. To address this issue, text augmentation was employed using the NLPAug library in Python, which generates synthetic data samples by introducing controlled linguistic variations. This study primarily utilized synonym replacement, where key tokens in minority-class messages (*spam*) were replaced with semantically similar alternatives drawn from a pre-trained language model, thus preserving message intent while increasing diversity. The data augmentation process used a synonym-based approach via the *nlpaug* library to generate exactly 1,183 new spam samples,

which were added to the original 747 spam messages. While the minority "spam" class was increased to a total of 1,930 samples (28.6%), the majority "ham" class remained unchanged at 4,825 samples (71.4%), resulting in a distribution that is more balanced but still retains a natural skew. The quality of these generated samples was validated through a programmatic comparison of message lengths and word counts, alongside manual verification of specific examples to ensure the original semantic meaning was preserved. No explicit feature selection or dimensionality reduction techniques (such as χ^2 filtering or mutual information-based selection) were applied to the Bag-of-Words and TF-IDF representations. This design choice was intentional, as both Naive Bayes and Random Forest incorporate intrinsic mechanisms that mitigate high dimensionality. Naive Bayes naturally downweights non-informative tokens through low class-conditional probabilities, while Random Forest performs implicit feature selection via random feature sub-sampling and split-based impurity reduction. Preserving the full feature space also supports interpretability by enabling direct inspection of influential lexical cues, which is central to the security-oriented analysis in this study.

Mathematically, given a spam message represented as a sequence of tokens $x_i = (w_1, w_2, \dots, w_T)$, synonym replacement generates a new message x'_i by substituting a subset of tokens:

$$x'_i = (w'_1, w'_2, \dots, w'_T), \text{ where } w'_j \sim S(w_j), \quad (9)$$

and $S(w_j)$ denotes the distribution of synonyms for token w_j . By iteratively applying this transformation to multiple spam samples, the size of the minority class was increased, leading to a more balanced dataset that improved classifier learning dynamics.

Following augmentation, the text messages were normalized through a series of cleaning operations aimed at reducing noise while preserving semantic content. This involved converting all characters to lowercase, removing extraneous whitespace, and eliminating punctuation, numeric characters, and non-alphabetic symbols that did not contribute meaningful features for classification. Additionally, common stopwords, such as "and," "the," and "of," were removed to reduce dimensionality and mitigate feature sparsity, while stemming or lemmatization was applied to collapse word variants into their canonical forms. These operations ensured that tokens with equivalent meanings were represented uniformly, reducing redundancy and improving model generalization.

After cleaning, feature extraction was performed using the Bag-of-Words (BoW) model, which transforms text into a high-dimensional numerical feature space. Each message x_i is represented as a sparse vector $v_i \in \mathbb{R}^d$, where d is the vocabulary size, and each component v_{ij} corresponds to the frequency of token j in message i . Formally, if t_j denotes the j -th token in the vocabulary, the BoW representation is defined as:

$$v_{ij} = \text{Count}(t_j, x_i), \quad (10)$$

where $Count(t_j, x_i)$ is the number of occurrences of token t_j in message x_i . To account for token importance, Term Frequency–Inverse Document Frequency (TF-IDF) weighting was also explored, where the weight of a token t_j is computed as:

$$TF - IDF(t_j, x_i) = TF(t_j, x_i) \cdot \log\left(\frac{N}{DF(t_j)}\right), \quad (11)$$

with $TF(t_j, x_i)$ representing the term frequency of t_j in message x_i , $DF(t_j)$ denoting the number of documents containing a token t_j , and N being the total number of messages. This weighting scheme effectively downscales the influence of common words and amplifies the significance of rare, discriminative terms, thus enhancing classification performance.

The binary classification setting necessitated encoding categorical labels into numerical form, with legitimate (*ham*) messages assigned a label of 0 and malicious (*spam*) messages assigned a label of 1. Formally, for each label $y_i \in \{ham, spam\}$, the encoding function $g: \{ham, spam\} \rightarrow \{0, 1\}$ is defined as:

$$g(y_i) = \begin{cases} 0, & \text{if } y_i = ham, \\ 1, & \text{if } y_i = spam. \end{cases} \quad (12)$$

This binary encoding not only simplifies the training process but also aligns with standard evaluation metrics such as precision, recall, and F1-score, which are optimized for binary classification tasks.

Through this rigorous preprocessing pipeline, the raw text dataset was transformed into a structured, balanced, and semantically meaningful numerical representation, forming the basis for effective model training. The integration of augmentation techniques ensured that the minority spam class was adequately represented, while the combination of cleaning, normalization, and BoW/TF-IDF vectorization produced sparse, high-dimensional features well-suited to the classification algorithms employed in this study. By reducing noise, addressing imbalance, and improving feature expressiveness, the preprocessing stage played a pivotal role in enhancing model robustness and optimizing classification performance in smishing detection.

3.4 Experimental setup

The experimental setup for this study was carefully designed to ensure reproducibility, scalability, and rigorous evaluation of the proposed machine learning framework for smishing detection. All experiments were conducted in a Python-based environment using Python version 3.10, with critical machine learning workflows implemented via the scikit-learn library, which provides robust and well-optimized implementations of supervised learning algorithms. Data augmentation was performed using the NLPAug library, a widely adopted natural language processing toolkit for generating synthetic text samples, to address the pronounced class imbalance

characteristic of SMS smishing datasets. From a technical perspective, NLPAug was selected because it applies linguistically constrained, token-level transformations such as synonym replacement and contextual word substitution that preserve the semantic intent of the original message while introducing controlled lexical variability. This approach is particularly suitable for short-form SMS text, where minor lexical variations can substantially improve model generalization and minority-class learning. Unlike vector-space oversampling techniques such as SMOTE, which are ill-suited to sparse textual representations and may generate semantically invalid samples, NLPAug operates directly on raw text and maintains syntactic plausibility, thereby reducing the risk of introducing noise into the training data. Additional libraries, including NumPy and Pandas, were used for data manipulation and analysis, while Matplotlib and Seaborn facilitated the generation of visualizations to examine class distribution, training behavior, and evaluation outcomes. The preprocessing pipeline, comprising text cleaning, tokenization, normalization, and feature extraction, was implemented using NLTK and scikit-learn’s feature extraction modules.

Model interpretability was derived directly from the intrinsic properties of the classifiers rather than post-hoc explainability techniques. For the Random Forest model, feature importance scores were obtained using the `feature_importances_` attribute, which measures the relative contribution of each lexical and structural feature to impurity reduction across the ensemble of decision trees; these importance rankings are presented in Fig. 6. For the Naive Bayes model, interpretability was achieved through analysis of class-conditional log probability ratios, which quantify the relative influence of individual tokens on spam versus ham classification decisions. Tokens with larger log probability differences exert a stronger influence on posterior class likelihoods, providing transparent insight into the model’s decision-making process; this token-level evidence is illustrated in Fig. 9.

Two machine learning algorithms, Naive Bayes and Random Forest, were selected for experimentation due to their complementary properties and proven effectiveness in text classification tasks. The Naive Bayes classifier was implemented using the Multinomial variant, which is particularly suited for discrete feature representations such as Bag-of-Words and Term Frequency–Inverse Document Frequency (TF-IDF) vectors. The model was trained with Laplace smoothing ($\alpha = 1.0$) to handle zero-frequency terms and ensure robust probability estimation, especially for tokens that may not appear in both training and testing subsets. The simplicity and computational efficiency of this probabilistic classifier made it an excellent baseline for comparison.

The Random Forest classifier was chosen as a more complex and robust ensemble-based method capable of capturing non-linear relationships in the data. The model was trained using 50 decision trees ($n_estimators = 50$) to ensure stability in predictions, with each tree limited to a maximum depth of 20 to avoid overfitting. The splitting criterion was set to Gini impurity, and a minimum of two samples was required to split an internal node. The number of features considered

at each split was set to the square root of the total number of features ($max_{features} = \sqrt{d}$), a common practice for optimizing Random Forest performance. Hyperparameter optimization for the Random Forest classifier was performed using an automated grid search strategy. The search space included the number of trees ($n_estimators \in \{50, 100, 200\}$), maximum tree depth ($max_depth \in \{10, 20, 30\}$), and the minimum number of samples required to split an internal node ($min_samples_split \in \{2, 5\}$). GridSearchCV with 5-fold stratified cross-validation was employed on the training set to evaluate candidate configurations. Model interpretability was obtained directly from the intrinsic properties of the classifiers. For the Random Forest model, feature importance was computed using the mean decrease in impurity (MDI), which quantifies the total reduction in Gini impurity contributed by each feature across all trees in the ensemble and is normalized such that the importance scores sum to one. For the Naive Bayes model, feature importance was derived from class-conditional log-probability ratios, computed as the difference between a token’s log posterior probabilities under the spam and ham classes, with larger absolute values indicating stronger discriminatory influence. The importance scores shown in Fig. 6 and 9 were obtained from the final trained models selected after hyperparameter optimization and are not averaged across cross-validation folds. For clarity of visualization, both figures present the relative ranking of the most influential features or tokens rather than their exact numerical values; however, all reported importance scores are normalized and directly comparable, reflecting the strength and direction of each feature’s contribution to the classification decision.

To ensure fair evaluation, the dataset was split into training and testing subsets using a stratified 80:20 split, preserving the original class distribution in both subsets. Stratified sampling was critical given the inherent imbalance in the dataset, as it guaranteed that both ham and spam classes were proportionally represented in the training and testing partitions. The training subset was used for model fitting and hyperparameter optimization, while the testing subset was exclusively reserved for final evaluation to prevent data leakage. Additionally, 5-fold stratified cross-validation was applied on the training data during hyperparameter tuning to assess model performance stability and prevent overfitting, ensuring that results generalized well across different partitions of the dataset.

The feature vectors were constructed using both Bag-of-Words and TF-IDF representations to capture both token frequency and discriminative importance. Each experiment was repeated using both representations to determine their respective impact on model performance. Furthermore, all numerical features were normalized using scikit-learn’s StandardScaler, where necessary to ensure consistent scaling across models, though this was primarily relevant to algorithms sensitive to feature magnitudes. The pipeline was orchestrated using scikit-learn’s Pipeline module, which streamlined preprocessing, feature extraction, and classification into a single reproducible workflow, reducing the potential for human error and ensuring that data transformation steps were consistently applied across training and testing phases. All computations were performed on a system equipped with an Intel Core i7 processor, 16 GB of RAM, and a Windows 10 64-bit

operating system. Although no GPU acceleration was required due to the relatively modest dataset size, the hardware environment provided sufficient computational power to run experiments efficiently, including model training, cross-validation, and hyperparameter tuning. The experimental design emphasized reproducibility by fixing random seeds during dataset splitting, model initialization, and augmentation to ensure consistent results across repeated trials.

3.5 Evaluation Metrics

Evaluating classification performance in the domain of smishing detection requires a set of metrics that not only measure overall correctness but also account for the dataset’s inherent class imbalance. Relying solely on overall accuracy can be misleading, as a classifier that predominantly predicts the majority class (ham messages) may achieve superficially high accuracy while failing to identify spam messages, which are the primary focus of this study. To provide a comprehensive assessment, this research employed Accuracy, Precision, Recall, F1-Score, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). Together, these metrics provide a holistic evaluation of classifier performance, focusing on both error distribution and discriminative ability.

Accuracy serves as a general baseline measure, quantifying the proportion of correctly classified messages relative to the total number of instances. In a binary classification setting with true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

While this metric offers a straightforward indicator of overall correctness, it does not differentiate between errors across classes and is therefore insufficient for highly imbalanced datasets like the one used in this study, where ham messages significantly outnumber spam messages.

To address this limitation, Precision and Recall were emphasized as complementary metrics. Precision measures the classifier’s ability to correctly identify spam messages among all instances predicted as spam, reflecting its capacity to minimize false positives. Mathematically, Precision is expressed as:

$$Precision = \frac{TP}{TP + FP}. \quad (14)$$

This metric is particularly relevant in mobile banking security, where false positives could erode user trust and system credibility. Conversely, recall (or True Positive Rate) quantifies the classifier’s sensitivity by evaluating its ability to correctly detect all actual spam messages, thereby minimizing false negatives:

$$Recall = \frac{TP}{TP + FN}. \quad (15)$$

High Recall is critical in smishing detection because failing to identify malicious messages can result in significant financial and reputational losses for mobile banking users.

The F1-Score provides a single composite measure that balances the trade-off between Precision and Recall. It is calculated as the harmonic mean of these two metrics:

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \quad (16)$$

Unlike Accuracy, which may be inflated by the predominance of ham messages, the F1-Score penalizes classifiers that perform well on one metric but poorly on the other, making it an essential measure of effectiveness for imbalanced classification tasks. In this study, F1-Score played a central role in evaluating the impact of data augmentation and model choice, offering a more balanced assessment of classifier robustness.

To further evaluate classification performance independent of threshold selection, the Receiver Operating Characteristic (ROC) curve and its associated Area Under the Curve (AUC) were utilized. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) across varying decision thresholds, providing insight into the model's discriminative power. The TPR and FPR are defined as:

$$TPR = \frac{TP}{TP + FN}, \quad (17)$$

$$FPR = \frac{FP}{FP + TN} \quad (18)$$

The AUC condenses this curve into a single scalar value between 0 and 1, with higher values indicating superior classification performance. Unlike Accuracy, AUC-ROC evaluates the model across all possible thresholds, making it highly relevant in production environments where decision thresholds may vary depending on the security sensitivity of the system.

In combination, these metrics provide a rigorous and contextually relevant framework for model evaluation. Accuracy serves as a broad performance indicator, while Precision emphasizes the reduction of false positives, and Recall prioritizes minimizing false negatives, which is critical in preventing smishing attacks. The F1-Score balances these competing priorities, and AUC-ROC offers a threshold-independent assessment of discriminative capacity. By applying this suite of metrics, the study ensures that both Naive Bayes and Random Forest classifiers are evaluated comprehensively, enabling informed conclusions about their effectiveness in detecting smishing in mobile banking communications.

4 Experimental results

4.1 Random forest

Fig. 4 and 7, are computed on the held-out test set obtained from the stratified 80:20 train–test split. As highlighted in Fig. 4, the Random Forest model demonstrates an accuracy of 0.983, a precision of 0.995, a recall of 0.946, and an F1-score of 0.969. This performance profile indicates a model that is highly conservative in minimizing false alarms while maintaining strong sensitivity to smishing. The near-perfect precision suggests that when the model identifies a message as spam, it is almost invariably correct, an essential characteristic in banking contexts where excessive blocking of legitimate SMS messages can undermine trust and increase support costs. The recall of 0.946 indicates that most attacks are intercepted on the first attempt. The F1-score, which represents the harmonic balance of precision and recall, confirms that the operating point is well-chosen following augmentation and preprocessing, rather than being merely an artifact of class imbalance.

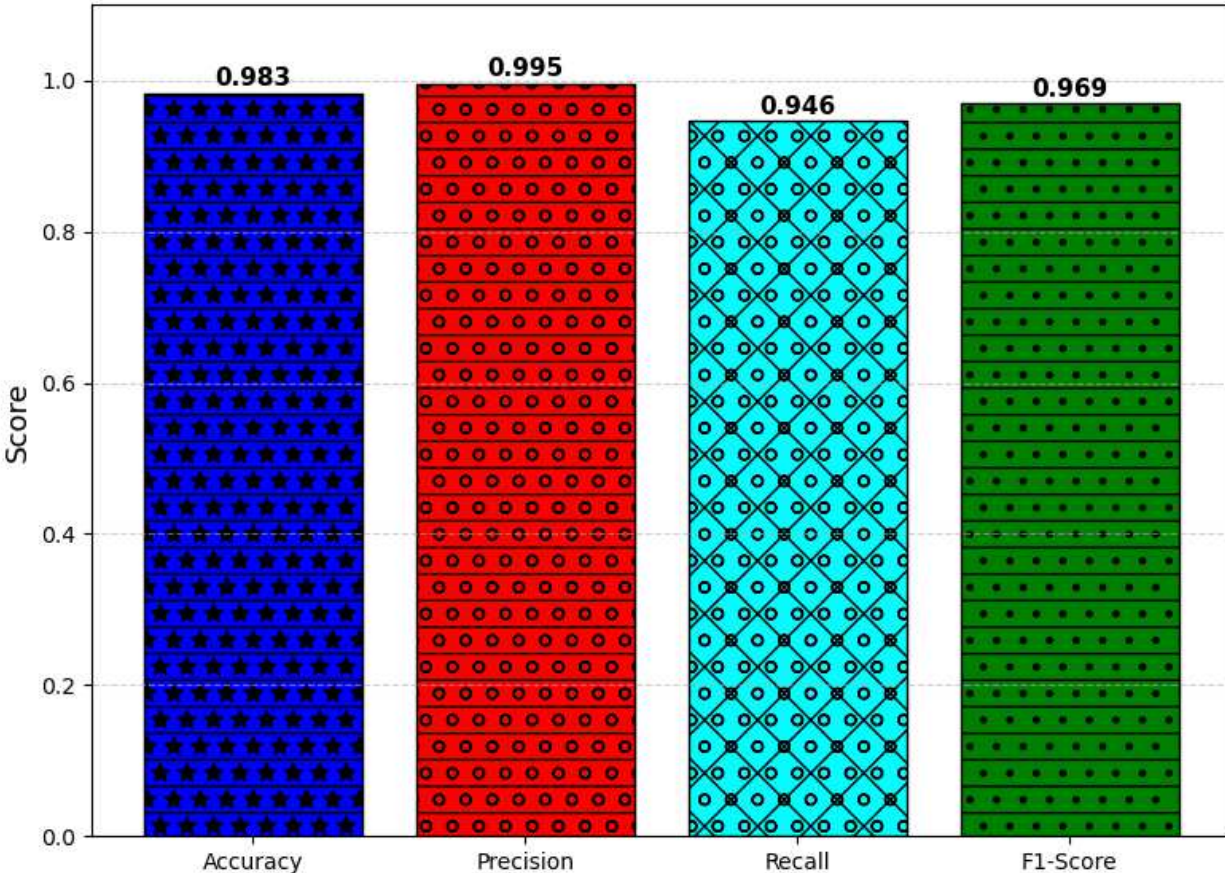


Fig. 4. Random Forest Performance Metrics

As highlighted in Fig. 5, the matrix elucidates the positioning of the bars: 963 instances of ham were correctly accepted, 2 instances of ham were incorrectly flagged (FP), 365 instances of spam

were correctly blocked (TP), and 21 instances of spam were missed (FN). These figures correspond to the primary metrics: Precision is 0.995, and Recall is 0.946; Specificity is also notably high at approximately 0.998. Operationally, the detector rarely inconveniences users, with only two benign messages being quarantined, and focuses its residual risk on the 21 false negatives, which likely suppress overt cues (such as few digits, minimal punctuation/emphasis) or closely mimic a transactional tone to resemble ham. If the policy prioritizes reducing misses further, a slight relaxation of the threshold or cost-sensitive training (penalizing FN more) could enhance Recall while accepting a controlled trade-off in Precision.

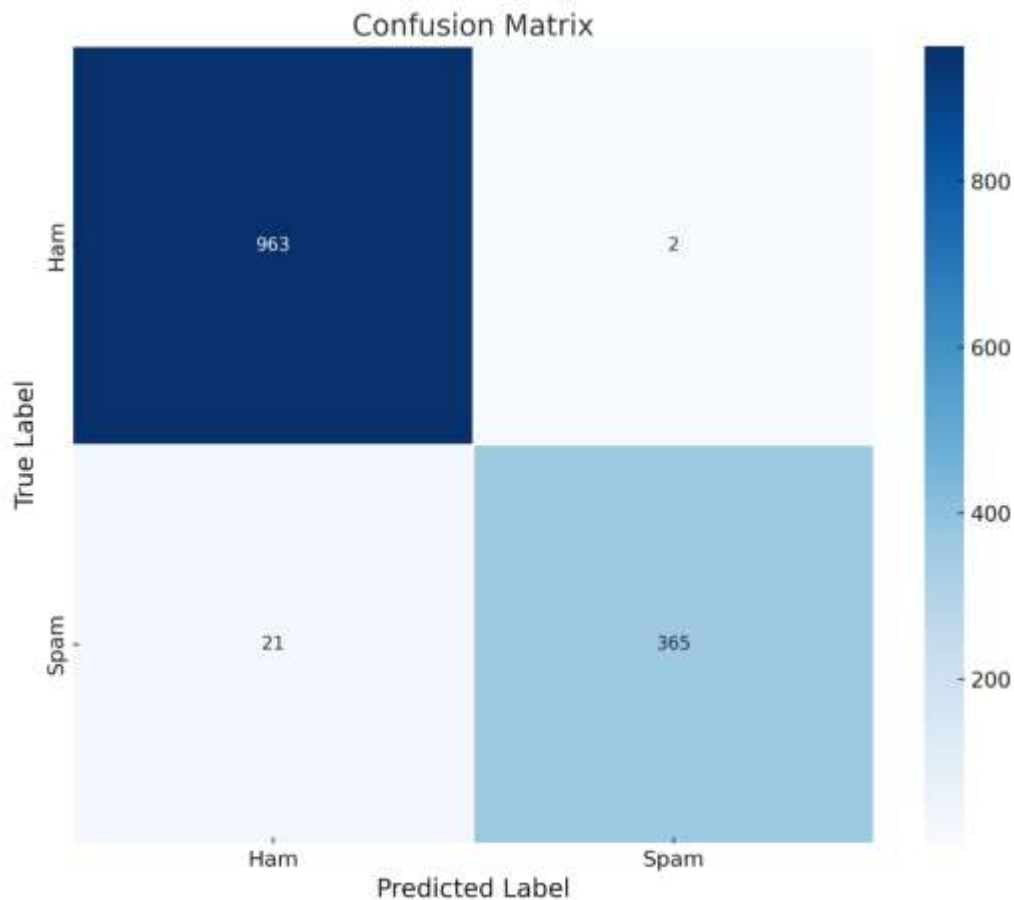


Fig. 5. RF Conf matrix

The confusion matrix shown in Fig. 5 corresponds to the held-out test set obtained from the stratified 80:20 train-test split. The integration of class-aware data preparation and ensemble learning bias is crucial. The use of NLP Aug oversampling has broadened the model's exposure to a wider range of spam phrasing, thereby mitigating majority-class bias. Additionally, normalization and token cleanup have effectively reduced noise, while BoW/TF-IDF has highlighted discriminative terms and minimized background vocabulary. The application of Random Forest's bagging and feature sub-sampling further reduces variance and identifies non-linear, conjunctive rules that are well-suited for text fraud patterns, such as the presence of

numerous digits, short length, and incentive keywords. The overall impact is illustrated in Fig. 4, where precision remains outstanding without a decline in recall.

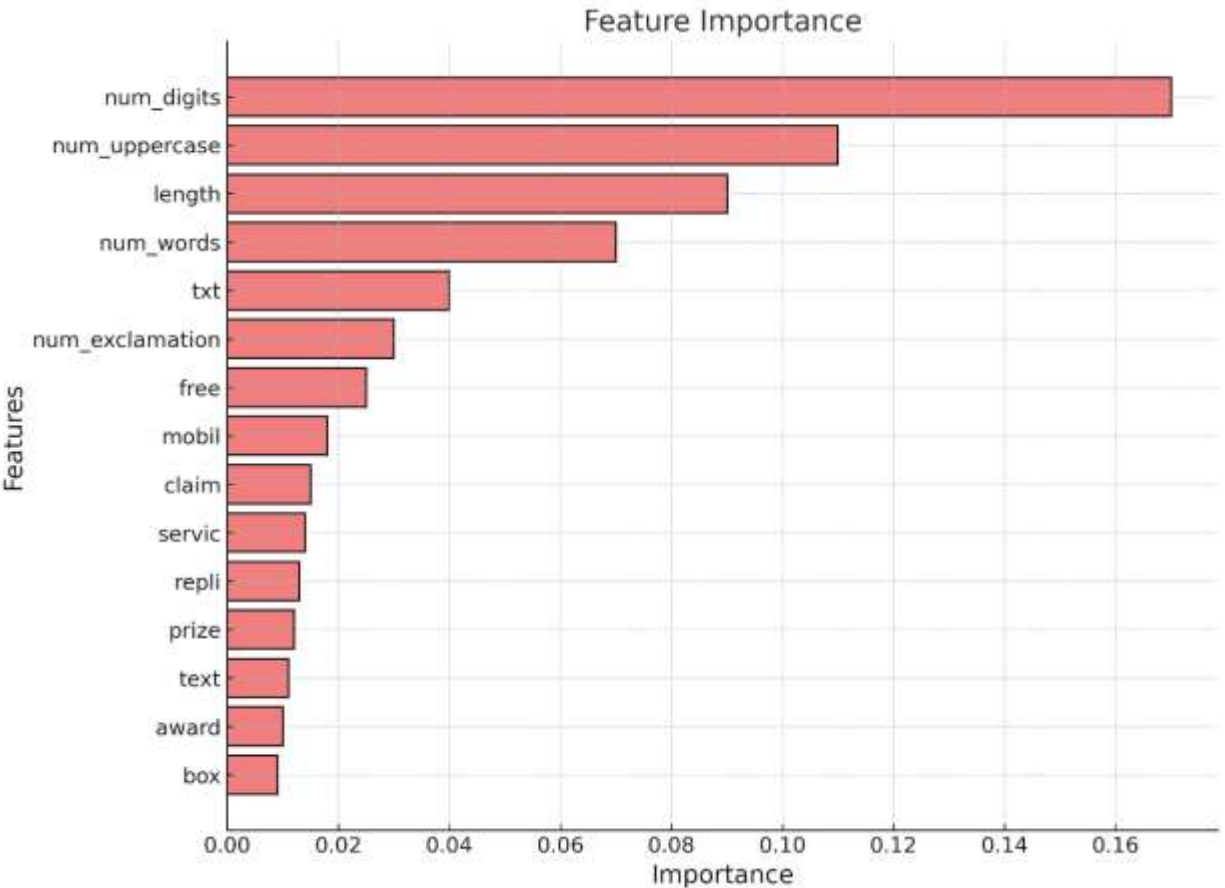


Fig. 6. RF Feature importance

Fig. 6 illustrates that the primary structural features, such as num_digits, num_uppercase, message length, num_words, and num_exclamation, significantly influence the split decisions. These are followed by token cues like “free,” “claim,” “prize,” “award,” “repli,” “servic,” “mobil,” “txt/text,” and “box.” These elements correspond with well-documented smishing signatures, including numeric payloads (codes, amounts, phone lines), emphatic typography (capital letters, exclamation marks), concise calls-to-action, and incentive-related lexicon. Importantly, the forest’s tree structure effectively captures interactions among these signals, which accounts for the high precision: it requires multiple spam-like cues in combination before blocking, thereby avoiding false positives on benign texts that contain only a single isolated cue. The false negatives depicted in Fig. 5 are consistent with low-signal smishing, characterized by neutral wording, few digits, no incentives, or vocabulary outside the emphasized token set sometimes indistinguishable from brief legitimate notifications. The two false positives likely result from benign promotions that coincidentally exhibit multiple spam-like traits (numbers, incentives, emphasis). Both behaviors are anticipated edge cases in SMS security and can be mitigated through threshold

tuning, URL/phone-pattern features, character n-grams to detect obfuscation, or a second-stage verifier (e.g., domain/brand reputation scoring). Random Forest serves as a primary filter in a mobile-banking pipeline, preserving user experience through minimal false positives while effectively capturing most attacks. Where an institution’s risk posture necessitates maximal recall, the decision threshold can be incrementally adjusted, or cost-sensitive/weighted training can be adopted, while monitoring precision. Enhancing features with URL/phone detectors, character-level n-grams, token entropy, and brand lexicons will specifically target the subtle, low-signal smishing that currently evades detection, further refining the confusion matrix without compromising the model’s core strength, precision grounded in interpretable, fraud-aligned signals. These results reflect the impact of NLPAug-based text augmentation, where the minority spam class was expanded using synonym replacement, contextual word substitution, and random insertion to improve lexical diversity without altering message intent.

4.2 Naïve Bayes performance

As illustrated in Fig. 7, the Multinomial Naive Bayes baseline achieves an Accuracy of 0.935, a precision of 0.892, a recall of 0.878, and an F1-score of 0.885. This performance profile suggests a balanced and reliable classifier that is competitive for large-scale text screening. The precision-recall trade-off is well-calibrated: a precision slightly below 0.90 indicates that most messages identified as spam are indeed malicious, while a recall close to 0.88 demonstrates the model’s capability to detect many smishing attempts. These results align with the Naive Bayes model’s inductive bias towards sparse, high-dimensional term-frequency features: the additive log-likelihood over tokens favors documents containing multiple high-evidence terms, and the Laplace smoothing applied during training stabilizes estimates for less frequent vocabulary introduced by augmentation. Fig. 7 also, reflects a robust baseline that is fast, interpretable, and effective, although it may not be as conservative regarding false alarms or as sensitive to subtle attacks as an interaction-aware ensemble.

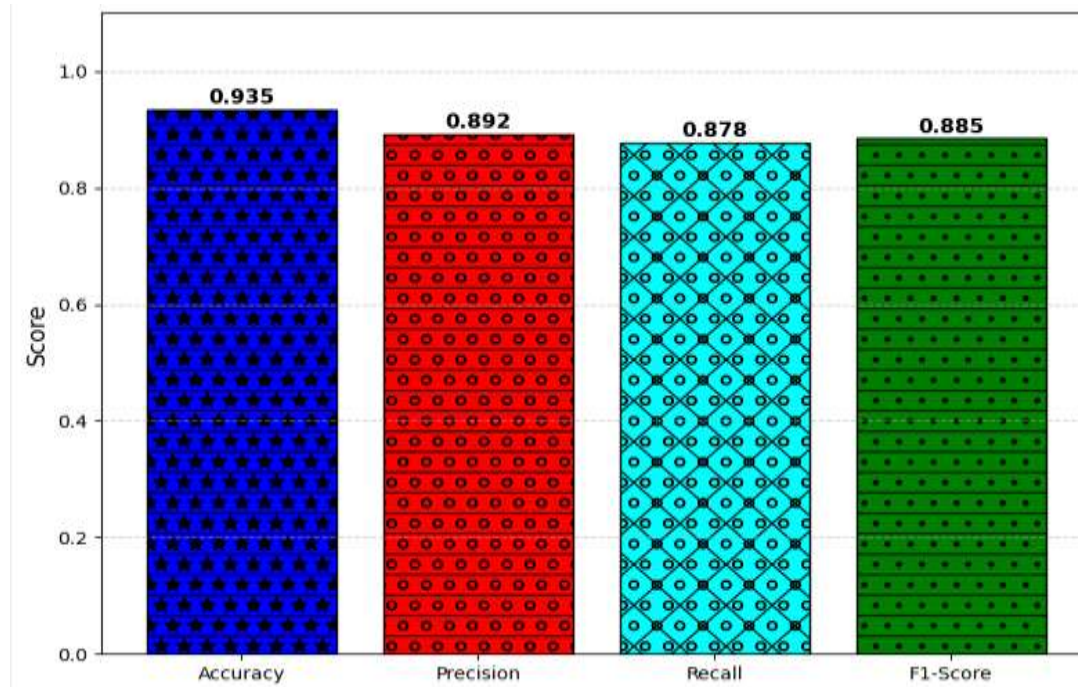


Fig. 7. Naive Bayes performance

The confusion matrix elucidates the underlying mechanics of the data: 924 ham messages are accurately accepted (true negatives), and 339 spam messages are correctly blocked (true positives). Conversely, 41 ham messages are erroneously blocked (false positives), and 47 spam messages are not intercepted (false negatives). These figures correspond to the primary metrics (Precision = 0.892; Recall = 0.878) and suggest a specificity of approximately 0.958, as highlighted in Fig. 8.

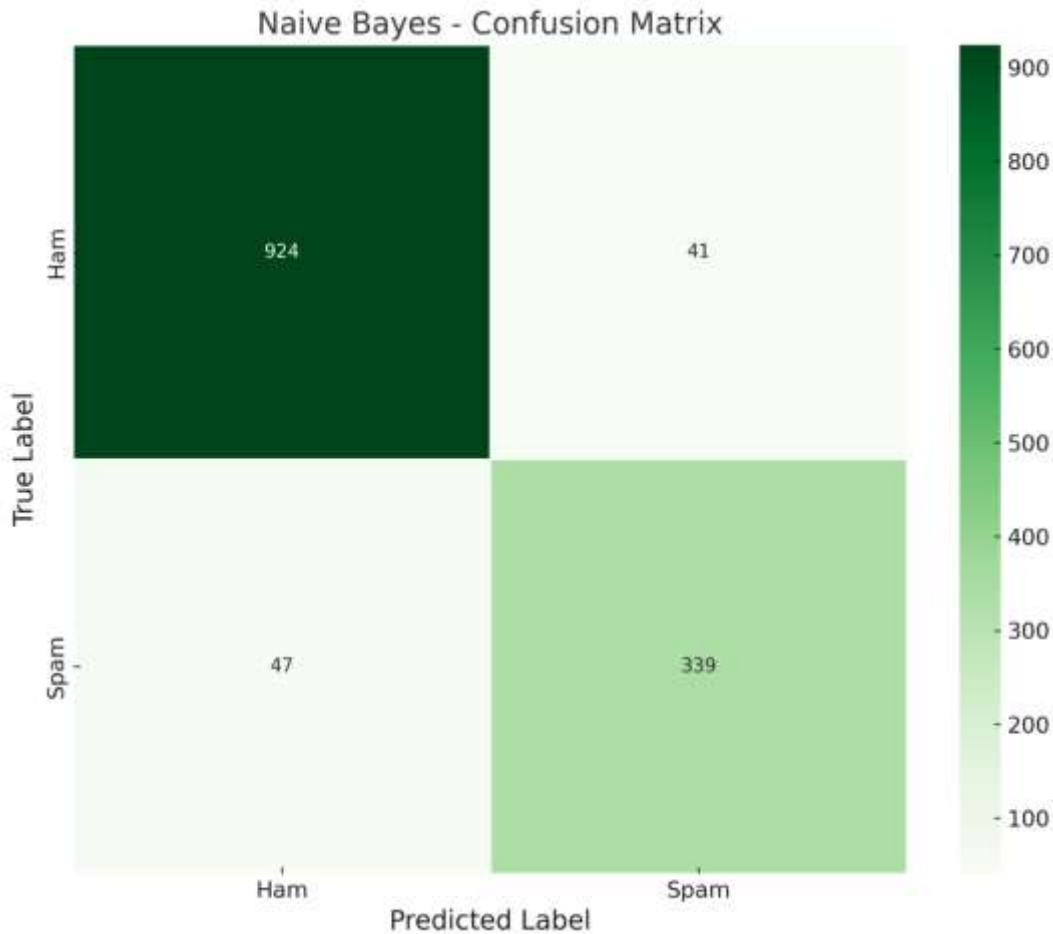


Fig. 8. Naive Bayes confusion matrix

The confusion matrix presented in Fig. 8 corresponds to the held-out test set obtained from the stratified 80:20 train–test split. Naive Bayes independently aggregates token evidence, which can result in the excessive penalization of legitimate promotions or notifications that contain several words commonly associated with spam, leading to 41 false positives. Conversely, it may fail to detect smishing attempts that either avoid explicit cues or obscure them, resulting in 47 false negatives. Messages that are concise, neutral in tone, and contain few numbers may be misclassified as non-spam under the Naive Bayes independence model. In contrast, legitimate offers containing incentive-related vocabulary may be incorrectly classified as spam, even when their context is benign. Adjusting the threshold can alter this balance; increasing recall typically results in additional false positives, while enhancing precision generally leads to an increase in false negatives.

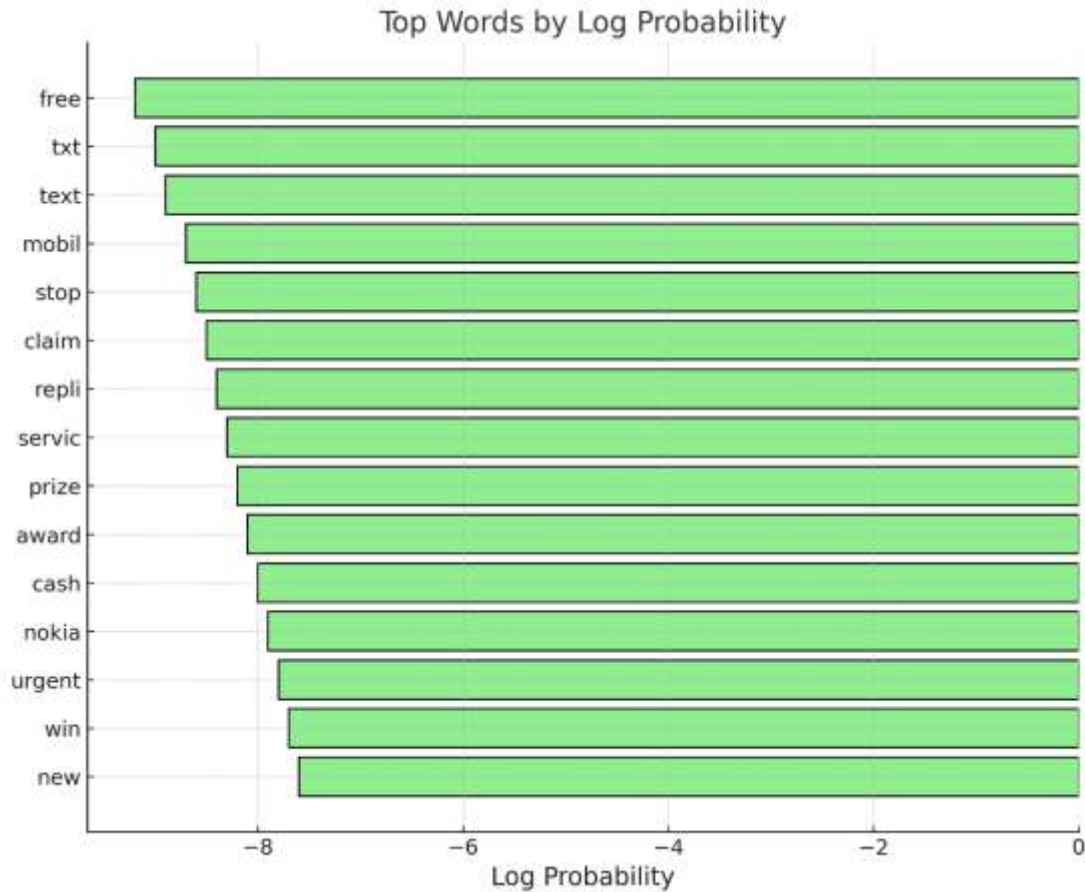


Fig. 9. Naive Bayes feature importance

The primary indicators of spam, such as "free," "txt/text," "mobil," "stop," "claim," "repli," "servic," "prize," "award," "cash," "nokia," "urgent," "win," and "new," elucidate both the strengths and residual errors of Naïve Bayes (NB). These tokens are recognized as canonical markers of smishing, encompassing incentive or lottery lexicon, urgency terms, service or response prompts, and device or brand hints. Given that NB aggregates token log-probabilities, any message containing multiple such words accumulates substantial spam evidence, resulting in high precision when these terms genuinely indicate attacks, as evidenced by the 339 true positives. However, terms like "stop" and "free" also appear in legitimate marketing and service communications; when they co-occur with other mild cues, the independence assumption of Naïve Bayes cannot discount the benign context, leading to observed false positives. Conversely, false negatives typically occur when adversaries craft messages devoid of these high-weight tokens, employ euphemisms, or rely on numeric or formatting cues (such as digit strings, capitalization, and punctuation) that Naïve Bayes, operating on plain Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF) unigrams, does not model as strongly. Naïve Bayes serves as a credible, interpretable baseline: it captures the dominant lexical signatures of smishing and converts them into stable, rapid decisions with solid F1 scores. Its limitations are principled: (i) the independence assumption

prevents learning nuanced interactions (e.g., numbers combined with punctuation and specific verbs), and (ii) its lexical focus renders it sensitive to benign text that shares spam vocabulary and blind to low-lexical-signal attacks. If operational priorities necessitate fewer misses, two low-cost refinements generally enhance NB: adopting Complement Naive Bayes, which is robust on imbalanced text, and enriching features with character n-grams and simple structural indicators (such as digit counts, capitalization, exclamation counts, and URL or phone patterns). If user trust requires fewer false alarms, probability calibration and threshold tuning should be applied, or Naïve Bayes should be gated behind a second-stage verifier (e.g., URL or brand reputation). Within these parameters, the results demonstrate that Naïve Bayes remains a robust and transparent first-line detector, whose behavior is well explained by its top tokens in Fig. 9, and is predictable and tunable for mobile-banking deployment.

An analysis of misclassified samples provides insight into the error characteristics of both models. False negatives predominantly correspond to smishing messages with low lexical signal, including very short or code-based texts (e.g., abbreviated or numeric-heavy messages such as “Gr8,” “2 win,” or “4U deal”) as well as messages employing unconventional slang, paraphrased wording, or stylistic expressions that are sparsely represented in the training data (e.g., “Go b4 him, God is gr8”). Such messages contain minimal overt spam indicators, limiting the effectiveness of unigram-based Bag-of-Words and TF-IDF representations. In contrast, false positives most commonly arise from legitimate messages that exhibit surface-level similarities to smishing, particularly transactional or organizational messages containing multiple digits, punctuation, or emphatic formatting (e.g., “Meeting at 10:30!!! Room 402” or “Your verification code is 483920”). These benign messages trigger structural and lexical features frequently associated with smishing, such as numerical patterns and punctuation emphasis, leading to occasional misclassification despite legitimate intent. Overall, the observed errors highlight the inherent limitations of word-level lexical features in capturing low-signal or obfuscated attacks and contextual intent, suggesting that future enhancements such as character n-grams, contextual embeddings, or secondary verification mechanisms could further reduce both false negatives and false positives without compromising interpretability.

5 Discussion

6.1 Model performance

In all key performance metrics, Random Forest (RF) demonstrates superior performance compared to Naive Bayes (NB): Accuracy 0.983 versus 0.935, Precision 0.995 versus 0.892, Recall 0.946 versus 0.878, and F1 0.969 versus 0.885. This disparity is anticipated, given RF’s capability to learn non-linear, conjunctive interactions among lexical and structural features (e.g., digits, punctuation, and incentive terms) through bagged decision trees. This approach reduces variance and overfitting while effectively capturing complex rules in sparse text spaces. Conversely, NB assumes conditional independence between tokens and aggregates their log-

likelihoods. This method performs well when strong spam tokens co-occur but may incorrectly flag legitimate messages containing a few "spammy" words and fail to detect obfuscated smishing.

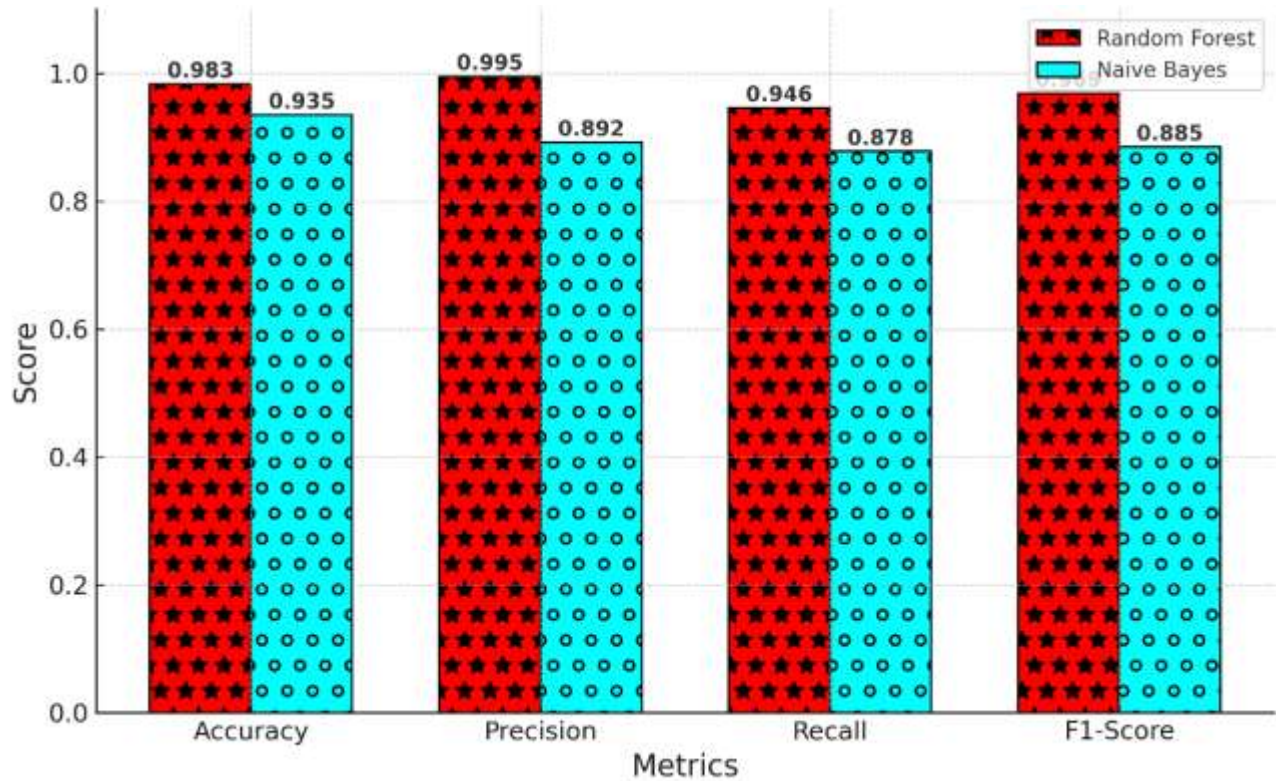


Fig. 10. Performance comparison of Random Forest vs Naive Bayes on the dataset

6.1 Threshold-independent discrimination

The ROC curves on the test set demonstrate an AUC of 0.998 for RF compared to 0.933 for NB, as illustrated in Fig. 11. This indicates that RF outperforms NB across decision thresholds and is likely to achieve higher true-positive rates for any given false-positive rate. Given that smishing detection involves class imbalance and varying operational costs of false negatives and false positives, it is advisable to utilize threshold-independent metrics such as ROC (and, where available, PR) to compare models before selecting an operating point.

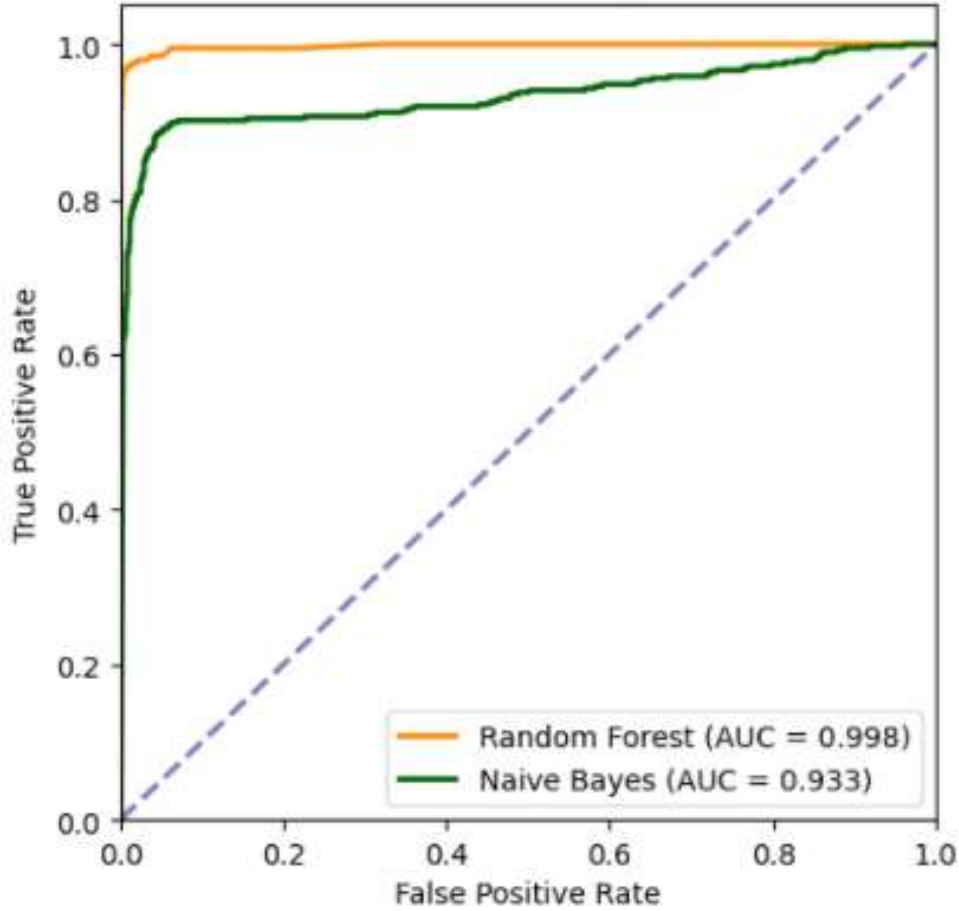


Fig. 11. ROC Curve comparison

6.2 Computational trade-off

Naive Bayes (NB) completes training in approximately 3.9 seconds, compared to approximately 42.5 seconds for Random Forest (RF), as highlighted in Fig. 12. This difference is attributed to NB's use of closed-form parameter estimates over sparse count features, whereas RF requires the fitting of numerous trees. In scenarios where frequent retraining is necessary, such as rapid drift adaptation or where computational resources are limited, NB serves as an appealing option as a fast baseline or initial filter. Conversely, in situations where security and user experience constraints are paramount, the additional training cost of RF is justified by its superior discrimination capabilities and error profile.

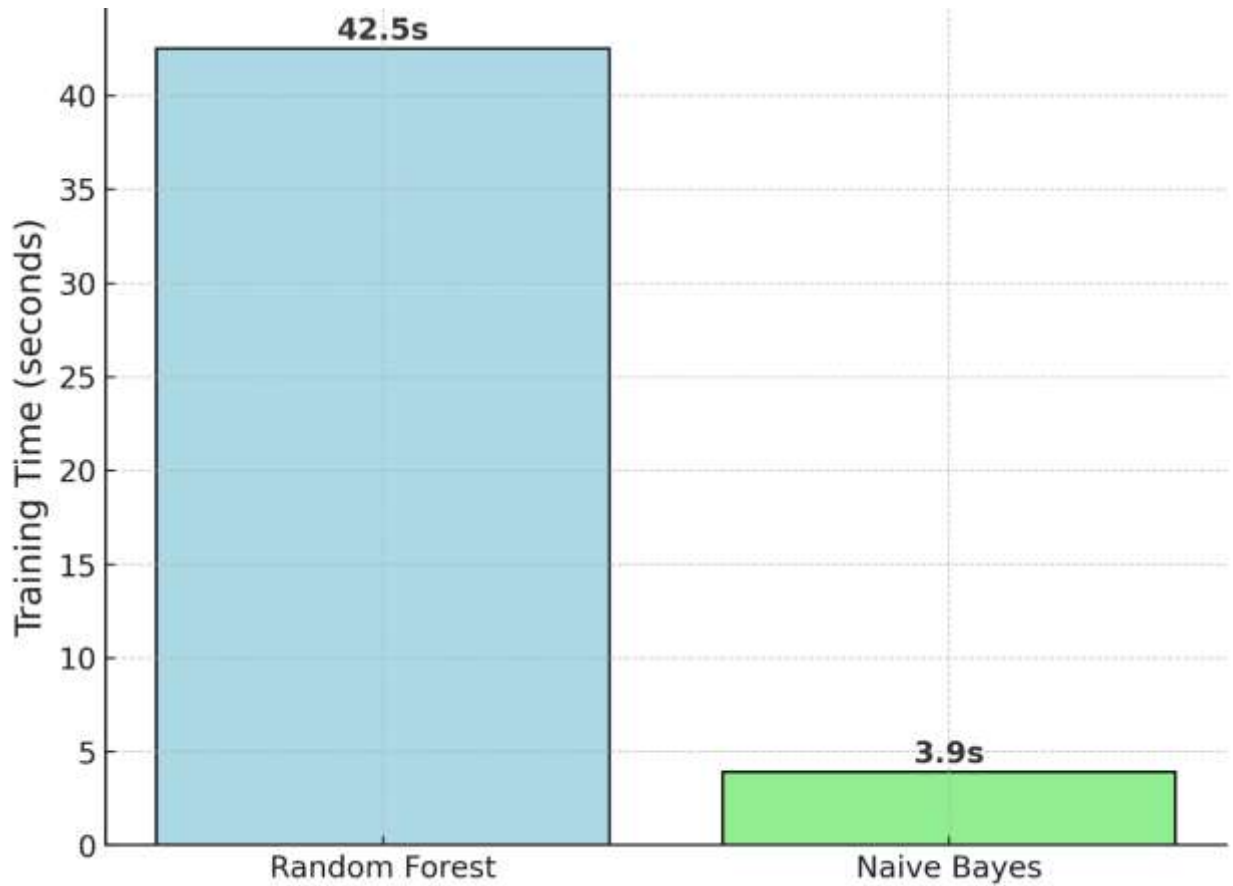


Fig. 12. Training time comparison of Random Forest and Naive Bayes measured on the training subset (80% split). Reported times correspond to model fitting only and exclude preprocessing and feature extraction.

6.3 Implications and recommended use

Based on the results obtained, the Random Forest (RF) model emerges as the more robust production classifier for mobile-banking smishing. This is attributed to its ability to leverage bagging and random subspace sampling, enabling it to learn non-linear, conjunctive interactions among lexical and structural cues with low variance. Consequently, it achieves high precision, strong recall, and excellent threshold-independent performance, as demonstrated by our ROC analysis, which aligns with foundational RF theory and practice [11]. Conversely, the Naive Bayes (NB) model remains valuable for rapid iteration and as a lightweight initial-stage filter. It is capable of training within seconds and performs effectively on sparse text; however, its conditional-independence assumption restricts its sensitivity to context-dependent or obfuscated attacks [10]. Both models benefited from the augmentation and preprocessing pipeline, with synonym-based text augmentation being widely recognized for enhancing generalization in imbalanced text classification [23]. or further improvements against low-lexical-signal smishing, the incorporation of character n-grams and URL/phone-pattern features is well-supported by prior security literature

[7]. Finally, to adjust the precision–recall trade-off in accordance with banking risk tolerance, it is advisable to apply probability calibration and thresholding techniques.

7. Conclusion

This study developed and evaluated a reproducible machine learning framework for smishing detection, demonstrating the superior detection performance of Random Forest alongside the computational efficiency and interpretability of Naive Bayes. By integrating linguistically informed text augmentation and consistent preprocessing within a unified pipeline, the framework effectively addressed class imbalance and achieved strong minority-class detection performance, highlighting the continued relevance of classical machine learning models for SMS-based security applications. Nevertheless, the study is subject to certain limitations. First, the reliance on a single publicly available dataset, while supporting reproducibility, may limit the generalizability of the findings to region-specific language use, cultural communication patterns, or emerging smishing tactics not represented in the data. Second, feature extraction was based on traditional Bag-of-Words and TF-IDF representations, which, although effective and interpretable, do not capture deeper semantic relationships or contextual dependencies within text. Future research should incorporate multi-regional or institution-specific datasets to enhance robustness across diverse linguistic environments. Additionally, integrating semantic or contextual language representations, such as word embeddings or lightweight transformer-based models, could improve the detection of low-lexical-signal smishing attempts. Model stability was assessed during hyperparameter optimization using stratified cross-validation, which produced consistent performance trends across folds, although confidence intervals or standard deviations are not explicitly reported. Future work may incorporate repeated resampling or bootstrap-based confidence intervals to further quantify metric variability and robustness across different data partitions. Finally, extending the framework toward real-time or deployment-oriented implementations would further strengthen its practical applicability within mobile banking systems.

Declaration of competing interest

Conflict of interest: The authors declare no conflicts of interest related to this study.

Data availability

Code and dataset available at: <https://github.com/KingsleyAmofo13/Sms-fraud-detection>

REFERENCES

- [1] Mishra S, Soni D (2023) Dsmishsms-a system to detect smishing sms. *Neural Computing and Applications* 35(7): 4975-4992. <https://doi.org/10.1007/s00521-021-06305-y>
- [2] Blancaflor EB, Cruz KJR, Monta FJC, Flores PE (2024) Unmasking the Threat: Analyzing and Mitigating SMS Smishing Attacks. In *International Congress on Information and Communication Technology* 69-78, Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-97-5441-0_7
- [3] Halim MI, Hasan MZ, Kabir MH, Hasan MN, Jaki H, Ahmad, Hasan H (2025) Enhancing Phishing Detection: A Machine Learning Approach to Predicting Malicious Emails, URLs, and SMS Messages. *Applied Computational Intelligence and Soft Computing* 2025(1): 6633979. <https://doi.org/10.1155/acis/6633979>
- [4] Jain AK, Gupta BB (2022) A survey of phishing attack techniques, defence mechanisms, and open research challenges. *Enterprise Information Systems* 16(4):527-565. <https://doi.org/10.1080/17517575.2021.1896786>
- [5] Mehmood MK, Arshad H, Alawida M, Mehmood A (2024) Enhancing smishing detection: A deep learning approach for improved accuracy and reduced false positives. *IEEE Access* 12:137176-137193. <https://doi.org/10.1109/access.2024.3463871>
- [6] Xu H, Qadir A, Sadiq S. Malicious SMS detection using ensemble learning and SMOTE to improve mobile cybersecurity. *Computers & Security* 2025; 154:104443. <https://doi.org/10.1016/j.cose.2025.104443>
- [7] Kytidou E, Tsikriki T, Drosatos G, Rantos K (2025) Machine learning techniques for phishing detection: A review of methods, challenges, and future directions. *Intelligent Decision Technologies* 19(6): 4356-4379. <https://doi.org/10.1177/18724981251366763>
- [8] Uddin MA, Islam MN, Maglaras L, Janicke H, Sarker IH (2025) Explainable detector: Exploring transformer-based language modeling approach for sms spam detection with explainability analysis. *Digital Communications and Networks*. <https://doi.org/10.1016/j.dcan.2025.07.008>
- [9] Salman M, Ikram M, Kaafar MA (2024) Investigating evasive techniques in sms spam filtering: A comparative analysis of machine learning models. *IEEE Access* 12: 24306-24324. <https://doi.org/10.1109/access.2024.3364671>
- [10] McCallum A 1998 A comparison of event models for naive Bayes text classification.
- [11] Breiman L (2021) Random Forests *Mach. Learn.* 45 (1): 5–32. *ed.*
- [12] Aljofey A, Jiang Q, Qu Q, Huang M, Niyigena JP (2020) An effective phishing detection model based on character level convolutional neural network from URL. *Electronics* 9(9): 1514. <https://doi.org/10.3390/electronics9091514>

- [13] Alotaibi A, Rassam MA (2023) Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense. *Future Internet* 15(2): 62. <https://doi.org/10.3390/fi15020062>
- [14] Timko D, Rahman ML (2023) Commercial anti-smishing tools and their comparative effectiveness against modern threats. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks* 1-12. <https://doi.org/10.1145/3558482.3590173>
- [15] Al Saidat MR, Yerima SY, Shaalan K (2024) Advancements of SMS spam detection: A comprehensive survey of NLP and ML techniques. *Procedia Computer Science* 244: 248-259. <https://doi.org/10.1016/j.procs.2024.10.198>
- [16] Jamil M, Mihajloska Trpcheska H, Popovska-Mitrovikj A, Dimitrova V, Creutzburg R (2025) Advancing Image Spam Detection: Evaluating Machine Learning Models Through Comparative Analysis. *Applied Sciences* 15(11): 6158. <https://doi.org/10.3390/app15116158>
- [17] Kamath BP, Geetha M, Acharya UD, Singh D, Rao A, Rai S, Shetty R (2025) Comprehensive analysis of word embedding models and design of effective feature vector for classification of Amazon product reviews. *IEEE Access* 13: 25239-25255. <https://doi.org/10.1109/access.2025.3536631>
- [18] Li X, Ling CX, Wang H (2016) The convergence behavior of naive Bayes on large sparse datasets. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11(1): 1-24. <https://doi.org/10.1145/2948068>
- [19] Speranza DL (2025) Data Augmentation for Imbalanced Fund-Policy Classification in Financial NLP: A Comprehensive Evaluation (Doctoral dissertation, Politecnico di Torino).
- [20] Johnson JM, Khoshgoftaar TM (2019) Medicare fraud detection using neural networks. *Journal of Big Data* 6(1): 63. <https://doi.org/10.1186/s40537-019-0225-0>
- [21] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16: 321-357. <https://doi.org/10.1613/jair.953>
- [22] Toledo JS, Segura-Bedmar I (2025) HULAT_UC3M at MiSonGyny 2025: Detecting Misogyny in Song Lyrics with Transformer Ensembles and Instruction-Tuned Generative Models. 2025. <https://doi.org/10.2139/ssrn.6153646>
- [23] Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N (2018) A survey on addressing high-class imbalance in big data. *Journal of Big Data* 5(1): 42. <https://doi.org/10.1186/s40537-018-0151-6>

- [24] Khan R, Venugopal A (2024) Exploring deep learning methods for text augmentation to handle imbalanced datasets in natural language processing. In 2024, the 3rd edition of IEEE Delhi Section Flagship Conference (DELCON) 1-8, IEEE.
<https://doi.org/10.1109/delcon64804.2024.10866632>
- [25] Stylianou N, Chatzakou D, Tsikrika T, Vrochidis S, Kompatsiaris I (2023) Domain-aligned data augmentation for low-resource and imbalanced text classification. In European conference on information retrieval 172-187, Cham: Springer Nature Switzerland.
https://doi.org/10.1007/978-3-031-28238-6_12
- [26] ElSabagh AA, Azab SS, Hefny HA (2025) A comprehensive survey on Arabic text augmentation: approaches, challenges, and applications. *Neural Computing and Applications* 37(10): 7015-7048. <https://doi.org/10.1007/s00521-025-11020-z>
- [27] Hasti P (2025) SMS phishing detection using machine learning and deep learning techniques.
- [28] Alshar'e M, Abualkishik A, Abuhmaidan K, Kayed A (2024) Enhancing Project Security: Unveiling Trust, Interpretability and Explainability in the Age of AI. In *5G Enabled Technology for Smart City and Urbanization System* 163-178, Chapman and Hall/CRC.
<https://doi.org/10.1201/9781003467892-12>
- [29] Nespoli P, Papamartzivanos D, Mármol FG, Kambourakis G (2017) Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys & Tutorials* 20(2): 1361-1396.
<https://doi.org/10.1109/comst.2017.2781126>
- [30] Paleyes A, Urma RG, Lawrence ND (2022) Challenges in deploying machine learning: a survey of case studies. *ACM computing surveys* 55(6): 1-29.
<https://doi.org/10.1145/3533378>