ISSN 2347 - 3983
Volume 13. No.7, July 2025
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter011372025.pdf
https://doi.org/10.30534/ijeter/2025/011372025

# SMS Spam Detection Using Machine Learning: An Experimental Study

**Sheela B[1], Komala R[2]**
[1]Ramaiah Institute of Technology, India, sheelabalu003@gmail.com
[2]Assistant Professor Department of MCA, India, komalar@msrit.edu

Received Date: May 28, 2025 Accepted Date: June 25, 2025 Published Date : July 07, 2025

**ABSTRACT**

The exponential growth of mobile communication has intensified the threat of SMS spam, compromising user security and trust in messaging platforms. This study addresses this challenge by designing and deploying a robust spam detection system using machine learning. We analyze a publicly available SMS dataset through rigorous pre-processing, including text normalization, tokenization, and feature engineering, followed by TF-IDF vectorization. A comparative evaluation of 11 classifiers—spanning probabilistic models, ensemble methods, and linear classifiers—reveals that ensemble techniques outperform traditional algorithms. The Extra Trees Classifier and XGBoost achieve state-of-the-art results, with 97.9% accuracy and 97.5% precision, demonstrating their efficacy in distinguishing spam from legitimate messages. To bridge the gap between research and practical application, we develop an interactive Streamlit web application that enables real-time spam classification with a user-friendly interface. This work underscores the potential of ensemble learning for text classification tasks and provides a scalable framework for combating SMS spam in real-world scenarios.

**Key words:** Ensemble Learning, Real-Time Classification, SMS Spam Detection, Streamlit Application, TF-IDF Vectorization

## 1. INTRODUCTION

Short Message Service (SMS) remains a vital mode of communication, facilitating both personal and professional interactions globally. However, the widespread adoption of SMS has made it a prime target for malicious actors who exploit the platform to disseminate spam—unsolicited messages that often include advertisements, fraudulent schemes, or phishing attempts. The persistent influx of such spam messages not only disrupts user experience but also poses significant security threats, including privacy breaches and financial losses.

Traditional spam detection methods, such as rule-based or keyword-matching filters, have proven increasingly ineffective as spammers continuously adapt their strategies to evade detection. These static approaches struggle to cope with the evolving and diverse nature of spam content. In contrast, machine learning offers a more adaptive and intelligent solution by enabling systems to automatically learn from historical data, recognize complex patterns, and generalize to previously unseen spam tactics.

This work is dedicated to the development of a comprehensive SMS spam detection system leveraging advanced machine learning techniques. The work encompasses the collection and preprocessing of real-world SMS data, the extraction and engineering of informative features, and the rigorous evaluation of multiple machine learning algorithms. Furthermore, to ensure practical applicability, we have designed and implemented an interactive web application that empowers users to classify SMS messages in real time. By integrating robust analytics with user-centric design, this work aspires to contribute an effective and scalable solution to the ongoing challenge of SMS spam.

## 2. PROPOSED METHODOLOGY

### 2.1 Objectives of Proposed Methodology

The primary objectives of this study are as follows:

#### 2.1.1 Comprehensive Data Preparation

To collect, pre-process, and analyse a real-world SMS dataset, ensuring the data is clean, consistent, and suitable for effective spam detection. This includes handling missing values, removing duplicates, and standardizing text data.

#### 2.1.2 Algorithm Implementation and Evaluation

To implement a diverse set of machine learning algorithms for the classification of SMS messages as spam or ham, and to rigorously compare their performance using relevant evaluation metrics such as accuracy, precision, recall, and F1-score.

#### 2.1.3 Identification of optimal Solution

To determine the most accurate and reliable algorithm(s) for SMS spam detection based on empirical results, with a focus on maximizing both accuracy and precision while minimizing false positives and false negatives**.**

#### 2.1.4Development of User-Friendly Application

To design and deploy an interactive web application that enables users to perform real-time SMS spam classification, thereby demonstrating the practical applicability of the developed models. The application should be accessible, responsive, and secure**.**

#### 2.1.5 Contribution to Research and Practice

To provide insights and recommendations for future work in SMS spam detection, contributing both to academic research and to the development of real-world anti-spam solutions. This includes discussing the scalability, adaptability, and limitations of the proposed system.

## 3. LITERATURE REVIEW

The field of SMS spam detection has witnessed significant advancements over the past two decades. Initially, detection techniques were primarily rule-based, relying on manually defined patterns, keyword filters, and regular expressions. While straightforward to implement, these static methods were prone to high false positive rates and quickly became ineffective as spammers adapted their strategies to bypass predefined rules [1].

The integration of machine learning (ML) marked a transformative phase in spam detection. One of the earliest and most influential models was the Naive Bayes classifier [2], favoured for its probabilistic foundation and effectiveness in handling text data. Its simplicity and speed made it a popular choice for early spam filters. As research progressed, more sophisticated algorithms such as Support Vector Machines (SVMs) **[3] and** Decision Trees were introduced, providing improved accuracy and generalization capabilities in varied spam scenarios.

In recent years, ensemble learning techniques have emerged as the leading approach in spam detection tasks. Algorithms like Random Forest [4], Gradient Boosted Decision Trees (GBDT), and XGBoost [5] aggregate predictions from multiple base learners, thereby enhancing model stability and reducing overfitting. Studies have reported that models such as Boosted Random Forest can achieve accuracy levels exceeding 98.47%, outperforming individual classifiers in both precision and recall [6].

Moreover, the advent of deep learning has introduced models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to the domain [7]. These architectures are capable of capturing complex contextual relationships in textual data. However, despite their superior learning capacity, deep learning models often require large datasets and high computational resources, limiting their applicability in lightweight or real-time spam detection systems [8].

Overall, the trend in the literature clearly demonstrates a shift from static, rule-based filters to adaptive, data-driven models, with ensemble methods currently offering the best trade-off between performance, complexity, and practical deployment.

## 4. DATASET AND PREPROCESSING

### 4.1 Dataset Description

For this study, we utilized the "SMS Spam Collection" dataset from the UCI Machine Learning Repository. The dataset contains 5,169 SMS messages labeled as either 'ham' (legitimate) or 'spam'. Figure 1 shows a sample of the SMS messages used in the dataset. Each entry consists of the message text and its corresponding label. The dataset is widely used in academic research and provides a reliable benchmark for spam detection studies. Table 1 summarizes key statistics of the SMS dataset.

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| 945 | ham | I cant wait to see you! How were the photos we... | NaN | NaN | NaN |
| 2641 | spam | You are guaranteed the latest Nokia Phone, a 4... | NaN | NaN | NaN |
| 5448 | ham | aight we can pick some up, you open before ton... | NaN | NaN | NaN |
| 4795 | spam | URGENT This is our 2nd attempt to contact U. Y... | NaN | NaN | NaN |
| 3322 | ham | She said," do u mind if I go into the bedroom... | NaN | NaN | NaN |
| 2640 | ham | Pandy joined 4w technologies today.he got job.. | NaN | NaN | NaN |
| 1819 | ham | Dunno dat's wat he told me. Ok lor... | NaN | NaN | NaN |
| 5024 | ham | I was gonna ask you lol but i think its at 7 | NaN | NaN | NaN |
| 2165 | ham | I'm not coming home 4 dinner. | NaN | NaN | NaN |
| 407 | ham | Hello! Good week? Fancy a drink or something l... | NaN | NaN | NaN |

**Figure 1:** Sample SMS Data from the Dataset

**Table 1:** Dataset Statistics

| Metric | Value |
|---|---|
| Number of messages | 5,572 |
| columns | 5 |
| Label distribution(ham/spam) | ~87%ham,~13%spam |
| Average characters/message | ~79 |
| Average words/message | ~18 |

Figure 2 presents a visual representation of the dataset statistics.

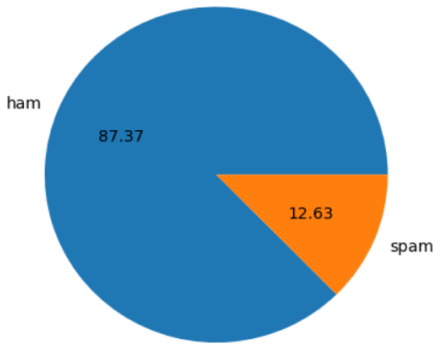


**Figure 2:** Dataset statistics

### 4.2 Preprocessing Steps

Effective preprocessing is crucial for accurate spam detection. The following steps were applied to the raw SMS data:

- Lowercasing: All text was converted to lowercase to ensure uniformity and reduce dimensionality.
- Tokenization: Messages were split into individual words (tokens) using the NLTK library.
- Removal of Non-Alphanumeric Characters: Only alphanumeric tokens were retained, removing punctuation and special symbols.
- Stopword Removal: Common English stop words were removed to focus on informative words.
- Stemming: Words were reduced to their root form using the Porter Stemmer to minimize redundancy.
- Feature Extraction: Additional features, such as the number of characters, words, and sentences in each message, were calculated to enhance model performance.

### 4.3 Data Transformation

A custom transform_text() function was implemented to automate the preprocessing pipeline, ensuring consistency and reproducibility. The function integrates all preprocessing steps and outputs clean, tokenized, and stemmed text suitable for vectorization.

## 5. MACHINE LEARNING ALGORITHM USED

This work evaluates the following machine learning algorithms for SMS spam detection. Each algorithm is briefly described below:

### 5.1 Naïve Bayes (NB)

A probabilistic classifier based on Bayes' Theorem, assuming independence among features. It is efficient and fast for text classification tasks, making it suitable for large-scale datasets.

### 5.2 Random Forest(RF)

An ensemble learning method that constructs multiple decision trees during training and outputs the mode of their predictions. It is robust to overfitting and effective for handling high-dimensional data.

### 5.3 Extra Tree Classifier (ETC)

An extension of Random Forest that adds extra randomness during tree construction. It typically improves accuracy and reduces variance by using random thresholds for splits, resulting in faster and often more accurate models.

### 5.4 Gradient Boosting Decision Tree (GBDT)

A sequential ensemble technique that builds models iteratively, where each new model corrects the errors of the previous one. Known for achieving high predictive performance in classification tasks.

### 5.5 Logistic Regression (LR)

A linear model for binary classification. It estimates the probability that a given input belongs to a particular class using a logistic function.

### 5.6 XGBoost

An optimized gradient boosting framework designed for speed and performance. It incorporates regularization to avoid overfitting and is widely used in competitive machine learning tasks.

### 5.7 Bagging Classifier (BgC)

An ensemble method that combines multiple instances of a base classifier trained on different random subsets of the training data. It helps reduce variance and improve model stability.

### 5.8 AdaBoost

An adaptive boosting technique that focuses on misclassified instances by adjusting their weights and combining weak classifiers to form a strong overall predictor.

### 5.9 Decision Tree (DT)

A non-parametric model that recursively splits the dataset into subsets based on feature values, creating a tree-like structure. It is easy to interpret and visualize but may over fit if not pruned.

### 5.10 K-Nearest Neighbor (KNN)

A non-parametric, instance-based learning method. It classifies new data points based on the majority label of the 'k' closest samples in the feature space.

### 5.11 Support Vector Classifier(SVC)

A powerful supervised learning model that finds the optimal hyperplane separating classes with maximum margin. It is effective in high-dimensional spaces but may be computationally intensive.

## 6. FEATURE EXTRACTION AND VECTORIZATION

Text data was transformed into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This approach captures the importance of words in each message relative to the entire dataset, enabling machine learning algorithms to process and learn from the content effectively. The TF-IDF representation reduces the impact of frequently occurring but less informative words, while highlighting rare but significant terms.

In addition to TF-IDF, feature engineering was performed to extract message-level attributes, such as message length, word count, and punctuation frequency, which can further aid in distinguishing spam from ham.

## 7. EXPERIMENTAL RESULTS

### 7.1 Evaluation Metrics

Model performance was evaluated using the following metrics, in line with IEEE best practices:

- Accuracy: Proportion of correctly classified messages.
- Precision: Proportion of messages classified as spam that are actually spam.
- Recall: Proportion of actual spam messages that were correctly identified.

- F1-Score: Harmonic mean of precision and recall.

Figure 5 illustrates the confusion matrix for the classification results.

**Table 2:** Confusion Matrix

| | **Predicted Ham** | **Predicted Spam** |
|---|---|---|
| Actual Ham | 971 | 14 |
| Actual Spam | 97 | 878 |

As illustrated by the confusion matrix in Table 2, the model's ability to distinguish between ham and spam messages is evident.

Figure 3 provides a visual representation of the top 30 most frequent words found in ham messages, offering insights into typical non-spam content.

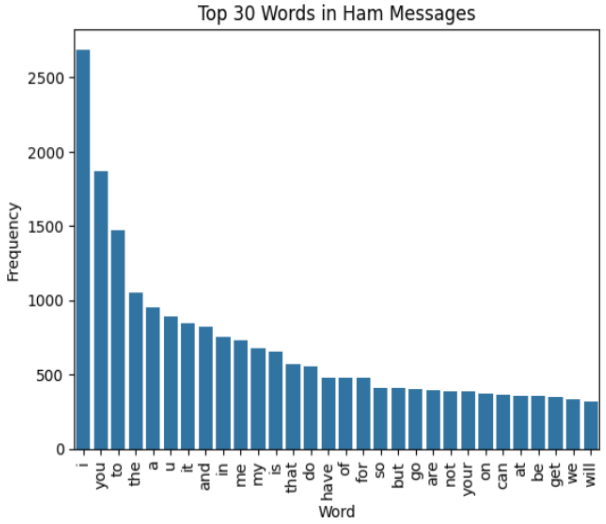


**Figure 3:** Top 30 words of Ham Messages

To contrast with ham messages, Figure 4 displays the top 30 words most frequently appearing in spam messages, which can aid in feature selection for classification.

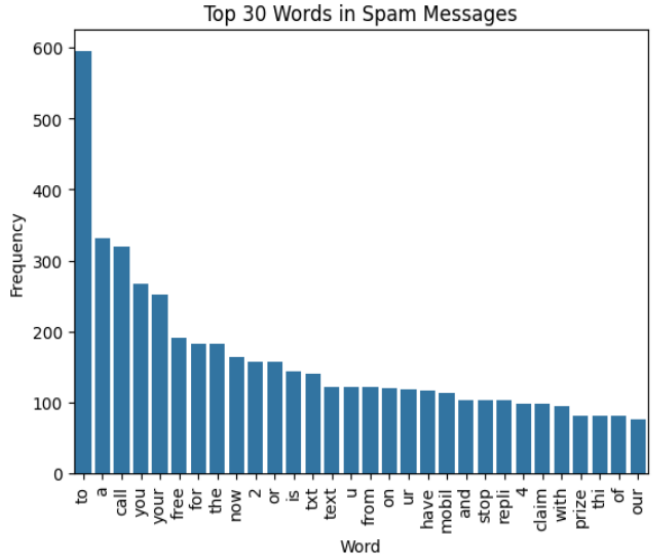


**Figure 4:** Top 30 words in Spam Messages

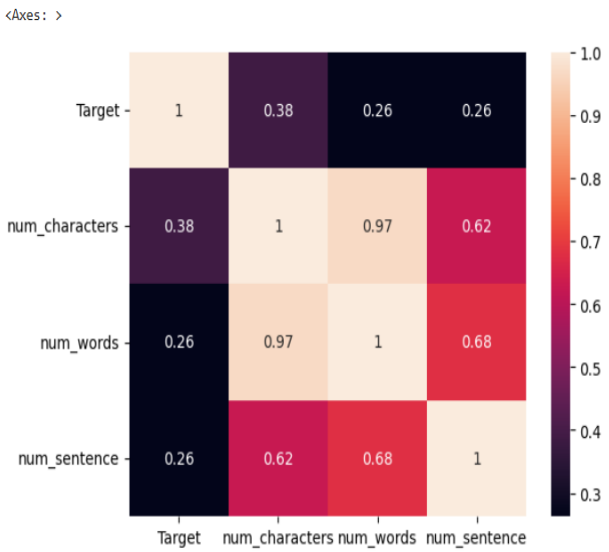


**Figure 5:** Confusion Matrix

**Table 3:** Classification Report

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 | 0.94 | 0.99 | 0.96 | 985 |
| 1 | 0.98 | 0.93 | 0.96 | 945 |
| Accuracy | | | 0.96 | 1930 |
| Macro Avg | 096 | 0.96 | 0.96 | 1930 |
| Weighted Avg | 0.96 | 0.96 | 0.96 | 1930 |

Table 3 shows the detailed classification metrics including precision, recall, and F1-score.

### 7.2 Performance Comparison

The classification performance of the implemented machine learning models was assessed using accuracy and precision. Figure 6 compares the accuracy and precision of different classifiers. Among all models, the Extra Trees Classifier (ETC) and XGBoost (XGB) achieved the highest accuracy (97.87%), while Naive Bayes (NB) and Random Forest (RF) showed perfect precision (1.0000), indicating zero false positives.

Ensemble methods generally performed better than individual models, highlighting their effectiveness in handling SMS spam data.

A visual comparison is provided in Figure 6.

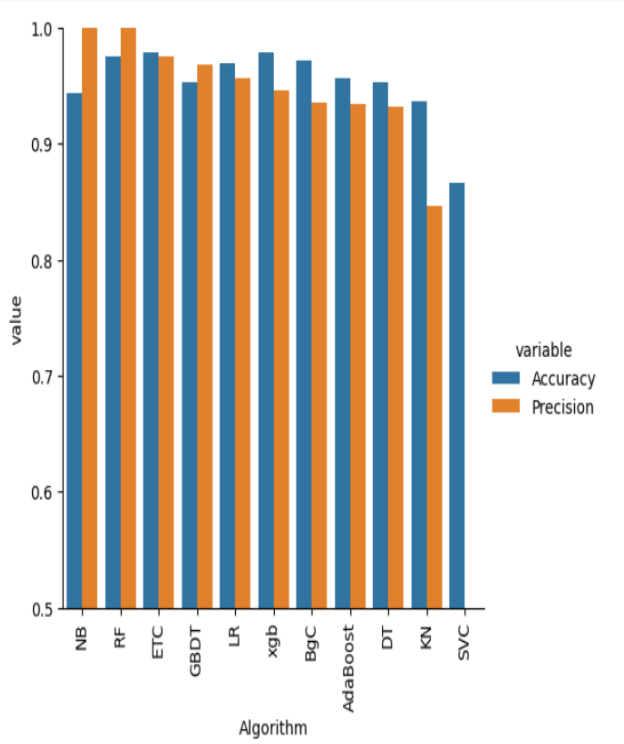


**Figure 6:** Accuracy and Precision Comparison of Classifiers

## 8. SYSTEM IMPLEMENTATION

### 8.1 Model Selection

Based on experimental results, ensemble methods such as Extra Trees and XGBoost were selected for deployment due to their superior performance, scalability, and robustness to overfitting.

### 8.2 Web Application

A user-friendly web application was developed using Streamlit. The application allows users to input SMS messages and receive real-time predictions (spam or ham) using the trained model.

**Key Features:**

- Real-time SMS spam classification
- Intuitive user interface with clear input/output
- Efficient preprocessing and model inference pipeline
- Secure handling of user data
- Scalability for integration into larger messaging platforms.

**Deployment and Public Access:**

- The SMS spam detection system was deployed as a web application on Streamlit Community Cloud, which provides a secure and scalable platform for hosting interactive machine learning applications. This deployment allows users worldwide to easily access the app via a public URL, submit SMS messages, and receive real-time spam classification results.
- The application not only demonstrates its practical utility but also encourages wider user engagement and feedback. The cloud-based deployment eliminates the need for local installation, making the tool readily available on any device with internet access.
- Furthermore, Streamlit's platform supports easy updates and maintenance, ensuring that the application can be improved continuously based on user input and evolving datasets.

Access the application here: https://sms-spam-detection-using--mll-publicurl.streamlit.app/

Figure 7 displays the interface of the deployed Streamlit web application.

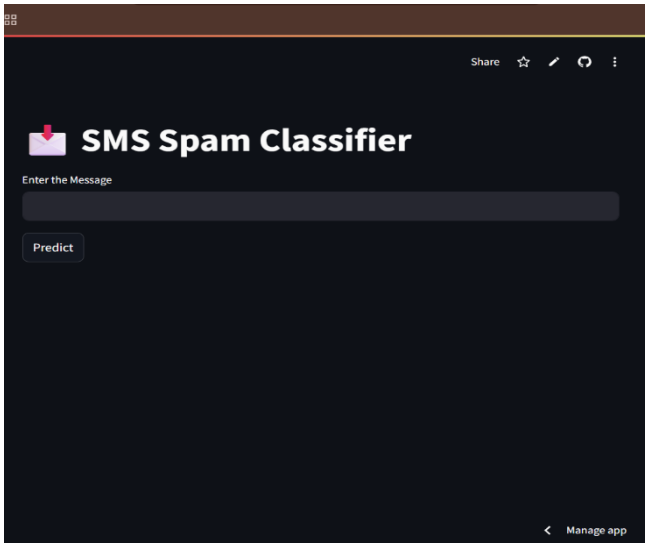


**Figure 7:** SMS Spam Classifier Web Application

## 9. DISCUSSION

### 9.1 Importance of Preprocessing

Comprehensive text preprocessing significantly improved model performance, especially for algorithms sensitive to noise and irrelevant features. The removal of stop words, stemming, and feature engineering contributed to higher accuracy and reduced false positives.

### 9.2 Ensemble Methods

Ensemble models such a Random Forest, Extra Trees, and Voting Classifier consistently outperformed individual classifiers, demonstrating the effectiveness of aggregating diverse model predictions. These methods are particularly robust to overfitting and can generalize well to unseen data.

### 9.3 Limitations

- The dataset is limited to English SMS messages; further work is needed for multilingual or code-mixed messages.
- Highly imbalanced datasets can challenge model performance, especially for rare spam types. Techniques such as SMOTE or cost-sensitive learning may be explored in future work.
- Real-world deployment may require additional considerations for privacy, latency, and integration with existing messaging infrastructure.

## 10. CONCLUSION

This study demonstrates the effectiveness of machine learning, particularly ensemble methods, for SMS spam detection. Extra Trees Classifier and XGBoost delivered the best results, achieving high accuracy and precision. The deployment of a web application showcases the practical utility of the approach. Future work may explore deep learning models, adaptation to multilingual datasets, and integration with real-time messaging systems.

## 11. PSEUDOCODE FOR KEY ALGORITHM

Based on results, the top-performing algorithms were:

- Extra Trees Classifier (ETC)
- XGBoost (Extreme Gradient Boosting)
- Random Forest (RF)
- Naive Bayes (NB)

**Pseudocode: Extra Trees Classifier (ETC)**

Input: Training dataset D with features X and labels y
Output: Trained Extra Trees Classifier model

1. For N trees in the ensemble:
   a. Randomly sample training data with replacement (optional)
   b. For each node in the tree:
      i. Select a random subset of features
      ii. For each feature, randomly choose a split threshold
      iii. Choose the best split based on information gain
   c. Grow the tree to full depth without pruning

2. For prediction:
   a. Pass input data through each tree
   b. Aggregate predictions using majority vote (classification)

Return: Aggregated prediction from all trees

**Pseudocode: XGBoost**

Input: Training data D = {(x1, y1), (x2, y2), ..., (xn, yn)}
Output: Trained XGBoost model

1. Initialize base prediction with a constant value (e.g., log odds)
2. For each boosting round t = 1 to T:
   a. Compute gradients and Hessians for current predictions
   b. Fit a decision tree to the gradients
   c. Compute leaf weights to minimize loss
   d. Update predictions:
      pred = pred + learning_rate * tree_output
3. Final prediction = sign(pred)

Return: Ensemble of T trees with learned weights

**Pseudocode: Random Forest**

Input: Dataset D with features X and labels y
Output: Random Forest model

1. For each of N trees:
   a. Create a bootstrap sample of D
   b. Build a decision tree:
      i. At each split, randomly select a subset of features
      ii. Choose the best split using Gini index or entropy
      iii. Recursively split until max depth or pure leaf

2. For classification:
   a. Each tree votes for a class
   b. Use majority voting to determine final output

Return: Majority vote across all trees

**Pseudocode: Naive Bayes**

Input: Training data with features (words) and labels (spam/ham)
Output: Trained Naive Bayes classifier

1. For each class c in {spam, ham}:
   a. Calculate prior P(c) = count(c) / total_messages
   b. For each word w:
      i. Count frequency of w in class c
      ii. Calculate likelihood P(w|c) with Laplace smoothing

2. For prediction on new message m:
   a. For each class c:
      i. Compute score: log(P(c)) + Σ log(P(w|c)) for each word w in m
   b. Predict class with highest score

Return: Class label (spam or ham)

## REFERENCES

[1] "SMS Spam Collection Dataset," UCI Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection.

[2] A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in *Proc. 11th ACM Symp. Document Engineering*, 2011, pp. 259–262.

[3] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos, "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages," in *Proc. 23rd Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, 2000, pp. 160–167.

[4] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[8] NLTK Project. [Online]. Available: https://www.nltk.org/. [Accessed: 28-May-2025].