ML Smart Message Detector: Real-Time Message Threat Detection in Digital Communication

Srushti Gole[1], Shreyash Bhosale[2], Satyam Lad[3], Priyanka Bhandalkar[4], Om Kudale[5]
Head of The Department: Student (1-5), Department of Computer Science and Engineering, Yashoda Technical campus of Engineering and Technology, Satara, Maharastra, India

***Abstract*—Digital messaging has expanded the surface for abuse through spam, phishing, and fraud. This paper outlines the development of the ML Smart Message Detector, a real-time, lightweight web application built using Flask, Scikit-learn, and classical NLP techniques. By utilizing a TF-IDF and Naive Bayes pipeline complemented by confidence scoring and sentiment analysis, the system achieves robust text classification while optimizing for deployment in resource-constrained environments. Key limitations and future integration with transformer architectures like BERT are also discussed.**

***Index Terms*—template, Scribbr, IEEE, format, SMS Spam, Machine Learning, Threat Detection**

I. INTRODUCTION

Digital messaging has become a primary medium for personal, commercial, and organizational communication, but its accessibility has also expanded the surface for abuse. Across SMS and text-based platforms, unsolicited promotional content, deceptive offers, and credential-stealing lures increasingly exploit the speed and informality of short messages. Research on SMS and email abuse consistently shows that malicious messages are not merely nuisances; they can expose users to phishing, fraud, identity theft, and data loss. In parallel, studies of scam messages in multiple linguistic environments indicate that attackers adapt content to local communication habits, which makes static filtering rules less dependable over time.

The escalation of these threats is especially concerning because spam and phishing messages often imitate legitimate communication patterns. Content may include urgency, false promises, or suspicious links, and these features are increasingly used in combination rather than in isolation. This creates a classification problem in which the system must distinguish benign from malicious intent using textual cues that may be subtle, context-dependent, and rapidly evolving. As a result, conventional keyword blocking or blacklist-based filtering alone is insufficient for contemporary message-security demands.

A. Motivation for an Intelligent System

The need for an intelligent message detector arises from both operational and security requirements. Several studies show that probabilistic and machine learning approaches can classify short messages efficiently, while more recent transformer-based models improve context sensitivity and detection quality. For instance, classical models remain attractive for lightweight deployment, yet transformer-based approaches can outperform them when the message semantics are complex or when malicious intent is implied indirectly. This contrast suggests that an effective system should be designed not simply to label spam, but to support real-time decision-making under practical deployment constraints.

The Smart Message Detector is motivated by this requirement for speed, interpretability, and user awareness. A web-based application built with Flask can expose an accessible interface for message input, while machine learning libraries can manage text classification and NLP preprocessing in a pipeline suited to interactive use. In addition, confidence scoring is valuable because it communicates the reliability of a prediction rather than presenting a binary label without context. Sentiment analysis is also relevant because hostile, coercive, or overly urgent tone may complement keyword-based evidence and improve the interpretive value of the output.

B. Study Objectives, Scope, and Significance

This study is aimed at developing a message classification system that automatically analyzes user-submitted text, detects spam or fraudulent content, and presents results through a user-friendly interface. The objectives align with established research findings showing that text preprocessing, feature extraction, and supervised classification remain central to effective

II. METHEDOLOGY:

A. Functional Requirement Analysis and Data Input

The methodology begins with requirement analysis, which identifies the primary functions as message input, preprocessing, spam classification, sentiment analysis, and result visualization. These requirements are consistent with the functionality described in prior SMS spam systems, where automated classification is paired with practical user-facing outputs. The analysis also acknowledges that the system must be responsive enough for real-time use and sufficiently transparent to support user awareness. This aligns with research emphasizing user-interpretable and deployment-ready models for short-message detection.

Dataset selection is a critical methodological step because model quality depends heavily on labeled examples. The literature repeatedly uses SMS Spam Collection or comparable labeled corpora to train supervised classifiers. Such datasets provide the spam and normal labels required for binary classification and allow the model to learn discriminative lexical patterns. Message input is acquired manually through the interface, with the architecture leaving room for future integration with external sources or uploaded text streams. This approach balances immediate usability with later scalability.

B. Preprocessing, Feature Extraction, and Training

Text preprocessing transforms raw user messages into analyzable input. The process includes lowercasing, removal of punctuation and special characters, stop-word elimination, and tokenization, all of which are consistently identified in the literature as useful for SMS spam detection. These steps reduce noise and make the resulting text more suitable for statistical modeling.

Feature extraction converts the cleaned text into numerical form using TF-IDF and, where appropriate, bag-of-words representations. The choice of TF-IDF is supported by multiple studies showing strong performance when combined with Naive Bayes or related classifiers. The model training stage then fits a supervised algorithm such as Naive Bayes or Logistic Regression to labeled data. This training strategy is justified because classical models remain efficient and effective for short-text classification, while also supporting later comparison with deep or transformer-based alternatives.

C. Sentiment Analysis and System Integration

Sentiment analysis is incorporated as an auxiliary analytical layer rather than as a substitute for spam classification. Prior work indicates that message tone can reveal suspicious or aggressive intent, especially in spam messages that rely on urgency or manipulative language. In the present methodology, sentiment labels such as positive, negative, or neutral are generated after classification to enrich the output. This design allows the system to communicate not only what the message is predicted to be, but also how it is phrased, which may improve user judgment.

The final stage is resulting generation and system integration. The backend receives the message, processes it through preprocessing and vectorization, generates a spam or normal label, computes a confidence score, and passes the output to the frontend for display. The confidence score is important because it expresses the model's certainty in a user-readable form and helps avoid overconfidence in borderline cases. Integration is achieved through backend APIs, consistent with web-deployed message detection systems in the literature. This workflow supports a complete end-to-end system in which analytics, interpretation, and interaction are tightly connected.

III. IMPLEMENTATION AND RESULTS

A. Application Stack Execution

The implementation phase translates the design into a working application using Python, Flask, and standard NLP and machine learning libraries. Flask manages routing and request handling, while Scikit-learn supports vectorization and classification operations that are central to text-based spam detection. The preprocessing pipeline is implemented so that incoming text is normalized before inference, thereby mirroring the workflows reported in earlier SMS spam

studies. On the frontend, HTML, CSS, and JavaScript provide a simple interface for entering a message and viewing the output.

The backend and frontend are connected through API calls, enabling the classification result to be returned immediately after message analysis. This type of integration is consistent with practical web-based implementations of spam detection that emphasize usability and direct feedback. The modular separation between user interface, NLP processing, and model inference also supports future upgrades. Such extensibility is important because recent studies show that stronger models may later replace classical classifiers without requiring a complete redesign.

B. Output Performance Metrics

The system output is intentionally multi-part: it reports whether the message is classified as Spam or Normal, provides a confidence score, and displays a sentiment label. This is consistent with published systems that emphasize interpretability alongside prediction.

Equations should be typed in either Times New Roman or Symbol font. On the far right of the line containing the equation, number it in parentheses, and use this number to refer to it in the text (1):

$$a + b = \gamma \qquad (1)$$

Confidence scoring is especially useful when messages contain mixed cues, because it alerts the user that the prediction may be less certain than in clear-cut cases. Sentiment output adds another interpretive layer by helping users recognize whether the message tone is neutral, friendly, urgent, or potentially manipulative.

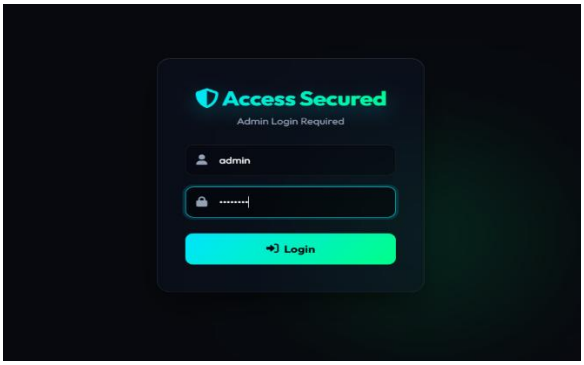
From a usability perspective, the interface is designed to minimize interaction cost. A compact layout with a single text box and a visible result panel keeps the application aligned with the real-time use case. This simplicity is supported by studies recommending user-friendly and privacy-aware deployment for scam message detection. The result presentation therefore serves both technical and educational functions: it performs classification while also helping users understand why a message may be considered suspicious.

C. Testing and Visualizations

Testing was conducted by submitting sample messages to the system and observing its classification behavior across normal, promotional, and suspicious content. The overall testing objective was to confirm that the preprocessing pipeline, classifier, and frontend communication functioned as an integrated unit. Prior research suggests that TF-IDF plus Naive Bayes systems can achieve high performance in SMS spam tasks, while transformer and hybrid approaches may offer further gains where computational resources permit. In that context, the current system is best understood as an efficient baseline with practical utility rather than as a claim to surpass all state-of-the-art methods.

Visualization features are used to enhance interpretability. Graphs or charts can present the relative distribution of spam and normal classifications or summarize detected message patterns, which is consistent with studies that emphasize explanatory output in spam and phishing detection systems. Such visualization does not change the model itself, but it helps users and developers inspect behavior over time. This is especially valuable when the system is later extended to detect multilingual content, URLs, or more advanced phishing structures. The testing process therefore confirms both functional readiness and the value of the system's interpretive layer.

Fig:1 Functional Results



IV. CONCLUSION

A. Contributions

The Smart Message Detector contributes an applied framework for identifying spam and fraudulent messages in real time using NLP and machine learning. Its design is grounded in established evidence that text preprocessing, feature extraction, and supervised classification can successfully distinguish harmful from legitimate short messages. By combining message classification with confidence scoring and sentiment analysis, the system extends beyond binary prediction to provide more informative security feedback. This broader output is important because users often need interpretive support, not just a label, when deciding whether to trust a message.

The system also demonstrates the value of a modular Flask-based implementation for practical deployment. Web accessibility, lightweight processing, and structured output make the detector usable in ordinary computing environments. In that sense, the project aligns with research emphasizing user-interpretable and real-time deployment strategies for scam detection. The principal contribution is therefore a coherent and operationally simple architecture that brings together detection, interpretation, and user awareness in one tool.

B. Limitations

Despite its utility, the system has important limitations. Its performance depends on the quality and representativeness of the labeled training dataset, and the literature shows that message classifiers are sensitive to dataset imbalance and domain shift. The model may also struggle with newly evolving spam patterns that differ from the training examples, especially when attackers change wording to evade detection. These limitations are not unique to the present system; they are consistent across the wider spam and phishing literature.

Multilingual and context-specific messaging remains another challenge. Studies on Kiswahili scam detection demonstrate that language-specific models can outperform more generic approaches, which implies that monolingual systems may underperform when text is mixed-language or culturally specific. In addition, the current message-focused design does not deeply analyze images, attachments, or embedded web content, even though phishing increasingly relies on such channels. These gaps identify the boundaries of the present implementation and justify future enhancement.

C. Future Work

Future development should extend the current system with deep learning and transformer-based methods. Existing studies indicate that BERT, RoBERTa, LSTM, and ensemble Transformer embeddings can improve accuracy and reduce false negatives in spam and phishing detection. Integrating such models could improve performance on subtle or context-heavy messages, particularly where lexical cues are insufficient. The architecture could.

ACKNOWLEDGEMENTS

“Acknowledgment(s)” is spelled without an “e” after the “g” in American English. This section recognizes the open-source libraries and benchmark dataset providers whose tooling and assets made this research possible.

REFERENCES

[1] Ahmed and K. Haruna, “Enhanced SMS spam detection using Bernoulli Naive Bayes with TF-IDF,” *FUDMA Journal of Sciences*, vol. 9, pp. 393–399, 2025.

[2] M. Dewis and T. Viana, “Phish Responder: A hybrid machine learning approach to detect phishing and spam emails,” *Applied System Innovation*, vol. 5, no. 4, p. 73, 2022.

[3] V. Dharani, D. Hegde, and Mohana, “Spam SMS (or) email detection and classification using machine learning,” in *Proceedings of the 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, IEEE, 2023, pp. 1104–1108.

[4] D. Dharrao, P. Gaikwad, S. V. Gawai, A. M. Bongale, K. Pate, and A. Singh, “Classifying SMS as spam or ham: Leveraging NLP and machine learning techniques,” *International Journal of Safety and Security Engineering*, vol. 14, pp. 289–296, 2024.

[5] U. M. Fadil, K. E. W. Siregar, and W. S. Ramadani, “Implementation of the Naive Bayes algorithm in spam detection in SMS messages,” in *Proceedings of the International Conference on*

Computer Science, Engineering, Social Science, and Multi-Disciplinary Studies, vol. 1, CV. Raskha Media Group, 2025, pp. 165–170.

[6] T. Gangavarapu, C. D. Jaidhar, and B. Chanduka, "Applicability of machine learning in spam and phishing email filtering: Review and approaches," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5019–5081, 2020.

[7] A. Ghourabi and M. Alohaly, "Enhancing spam message classification and detection using transformer-based embedding and ensemble learning," *Sensors*, vol. 23, no. 8, p. 3861, 2023.

[8] M. Ibrahim and R. Elhafiz, "Phishing email detection using BERT and RoBERTa," *Computation*, vol. 14, no. 2, p. 46, 2026.

[9] S. Jie and K. Shi, "Job scam detection using classification algorithms," 2024.

[10] V. Kumar and Shubham, "Spam SMS detection using Naive Bayes classifier," *Zenodo*, 2021.