International Journal of Intelligent Computing Systems
July-2026 | IJICS | Volume-2 Issue-2 | ISSN: 3107-7218

The Next Frontier In Digital Defence Cutting-Edge NLP For Spam Filtering

Chandu Sai Maheswari[1],Chandu Nagaraju[2]
Student[1], Assistant Professor[2]
Priyadarshini Institute of Technology & Science for Women, Chintalapudi, Andhra Pradesh, India[1, 2]

Abstract - In today's connected digital world, the way we communicate has changed a lot. Emails, text messages, and messaging apps are now important tools for both personal and business communication. But this growth also brings a new problem: spam messages. These unwanted messages fill up inboxes and can be a security and privacy risk. This project uses a machine learning method to classify spam messages. It uses natural language processing (NLP) and a simple, real-time interface built with Streamlit. We trained and tested several models using the publicly available SMS Spam Collection dataset, including Logistic Regression, Naive Bayes, and Support Vector Machine (SVM). We checked each model's performance using accuracy, recall, and F1-score, and then combined the best models into an interactive web application. Users can type in a message and get an instant result on whether it's spam. The app also shows key features, gives a spam confidence score, and includes interactive visualizations. The results show how effective it can be to use traditional machine learning together with a user-friendly design to address the common issue of spam messages in an efficient and clear way.

Keywords — Spam Detection, Machine Learning, TF-IDF, SMS Spam Dataset, Logistic Regression, Naive Bayes, SVM, NLP, Streamlit, Real-time Classification, Feature Importance, Confusion Matrix.

I. INTRODUCTION

In our connected world, how we communicate has changed a lot. Whether it's talking with friends or making business deals, digital tools like SMS, email, and messaging apps are part of our everyday lives. But with these tools comes a big problem: spam.

Spam used to be just poorly written emails trying to sell something.Now, it's much more advanced. Today's spam isn't just annoying—it can trick people into clicking on bad links, sharing personal information, or even sending money. So, spam has gone from being a small annoyance to a serious cybersecurity issue. That's why having good spam detection systems is more important than ever.

In the early days of email, simple methods like rule-based filtering and blacklists worked well.These systems checked for certain words or suspicious sender addresses. But the spam world has changed a lot. Spammers now use tricks like hiding text, making messages look like real ones, and changing their content quickly. So old rules can't keep up with the new ways spam is sent.

That's where machine learning comes in. Unlike traditional methods that follow fixed rules, machine learning learns from data. It can spot new trends, different message styles, and new spam tricks without needing to be told what to look for. With enough examples, a well-trained machine learning model can tell the difference between real messages and harmful ones quickly and accurately.

This project, called "The Next Frontier In Digital Defence Cutting-Edge NLP For Spam Filtering," is a real-world solution to this problem.It uses Natural Language Processing to look at message text and turn it into numbers using TF-IDF. These numbers are then used with machine learning algorithms like Logistic Regression, Naive Bayes, and Support Vector Machine to decide if a message is spam or not.

But being accurate isn't the only thing that matters.People need to trust the system, especially when it decides what they see. That's why this project also includes a simple web interface using Streamlit. Users can type in a message and see right away if it's spam or not. The system also shows a confidence score and highlights the parts of the message that made the model make its decision. This makes the system more useful and builds trust with the user.

Ultimately, this project is about more than just stopping spam.It's about creating a smart, easy-to-use system that changes with user needs, respects their trust, and keeps up with the ever-changing threats in the digital world.

II. METHODOLOGY

Building a reliable spam classification system requires more than just selecting a machine learning algorithm. It needs a well-thought-out pipeline that ensures accuracy, scalability, and real-time performance. This project takes a structured approach, moving from raw data handling to full deployment in a live, interactive application. The methodology can be divided into three main phases: data preprocessing, feature extraction, and model integration. Each phase is crucial for helping the system learn and perform effectively.

2.1 Data Preprocessing

Text data, especially from real-world messages, can be messy and inconsistent. Misspellings, extra spaces, and irrelevant symbols can make it hard to understand the true meaning of a message. That's why the first step in **Natural Language Processing (NLP)** is data cleaning.

In this project, the raw SMS messages from the SMS Spam Collection dataset went through several cleaning operations to prepare them for machine learning. These preprocessing steps help the model focus on the main message content instead of being distracted by noise.

Key Preprocessing Steps:
•**Lowercasing:** All messages were changed to lowercase to ensure that words like "FREE" and "free" were treated the same by the model.
•**Punctuation and Symbol Removal:** Unnecessary symbols, digits, and punctuation were removed, as they rarely add meaning in spam detection.
•**Stopword Elimination:** Common words like "is," "the," and "at" were removed using standard NLP stopword libraries. These words appear frequently in both spam and non-spam messages, offering little to no distinguishing power.
•**Optional Lemmatization:** In some cases, lemmatization was used to reduce words to their root forms (e.g., "winning" became "win"), ensuring consistency in feature representation.

This process resulted in a clean, standardized dataset, making sure that every word carried as much meaningful information as possible.

2.2 Feature Extraction using TF-IDF

Once we cleaned the data, the next challenge was converting the text into a numerical format that machine learning models can understand. Words alone do not have any mathematical meaning to an algorithm. To bridge this gap, we used the common technique called TF-IDF (Term Frequency-Inverse Document Frequency).

TF-IDF does not simply count how often a word appears in a message. It also assesses how important that word is by looking at how frequently it appears across all messages. Words that show up often in one message but rarely in others, like "win" or "urgent," get higher weights. This is a key feature in spam classification.

This scoring method lets the model focus on unique and significant terms that set spam apart from non-spam.

TF-IDF Configurations Used:
• **Unigram Model:** Captures individual words like "win," "claim," or "account." It helps find clear spam indicators.
• **Bigram Model:** Captures pairs of words that appear together, such as "free entry" or "claim prize." Bigrams give more context than single words and are especially useful for spotting spam phrases instead of just keywords.

The mix of unigrams and bigrams improves the model's ability to understand both individual word relevance and **contextual patterns**, making predictions more accurate and detailed.

2.3 Model Construction and Training

Once the message data was cleaned and transformed into numerical vectors using TF-IDF, the next step was to build machine learning models capable of accurately distinguishing between spam and legitimate (ham) messages. To ensure a balanced comparison and explore different learning behaviors, multiple classifiers were implemented and tested.

The following supervised learning algorithms were trained using the processed dataset:

Logistic Regression

A linear model that calculates the probability of a message being spam based on weighted input features. It's fast, interpretable, and well-suited for binary classification.

Multinomial Naive Bayes

A probabilistic model that assumes independence between features. Despite this simplification, Naive Bayes is renowned for its exceptional performance with text data, particularly when utilizing frequency-based inputs such as TF-IDF.

Support Vector Machine (SVM)

A powerful classifier that finds the optimal decision boundary between spam and ham classes in high-dimensional space. It is effective in handling sparse data, such as TF-IDF vectors.

Each model was evaluated independently to observe how different algorithms perform on the same task. These models bring different strengths—while Logistic Regression offers simplicity and speed, SVM provides a strong margin-based separation, and Naive Bayes handles word distributions naturally.

Training-Testing Split

To evaluate the generalization ability of each model, the dataset was divided into:

- **80% Training Data**
  Used to teach the model the patterns in spam and ham messages.
- **20% Testing Data**
  Used for evaluation on unseen messages to assess how well the model performs in real-world scenarios.

2.4 Model Evaluation and Visualization

Once the models were trained, they were subjected to thorough evaluation using multiple performance metrics to understand their effectiveness in spam detection.

Accuracy Score

Accuracy was calculated to determine the overall performance of each classifier. It measures the proportion of total correct predictions (both spam and ham) out of all predictions made. While useful, accuracy alone may not reflect model quality when class imbalance is present (e.g., more ham messages than spam).

Precision, Recall, and F1-Score

To better assess the model's spam-detection ability, the following metrics were used:

- **Precision**: Measures how many of the messages predicted as spam were actually spam. High precision means fewer false positives.
- **Recall**: Measures how many actual spam messages were correctly identified. High recall means fewer false negatives.
- **F1-Score**: The harmonic mean of precision and recall. It provides a single score that balances both metrics, especially important when the cost of misclassification varies.

These metrics are particularly useful in spam detection, where **false positives** (legitimate messages marked as spam) and **false negatives** (spam messages that slip through) have different implications.

Confusion Matrix Visualization

To further interpret the model's performance, **confusion matrices** were generated for each model using **Seaborn**, a Python visualization library. The confusion matrix shows:

- **True Positives** (spam correctly identified)
- **True Negatives** (ham correctly identified)
- **False Positives** (ham wrongly marked as spam)
- **False Negatives** (spam messages missed)

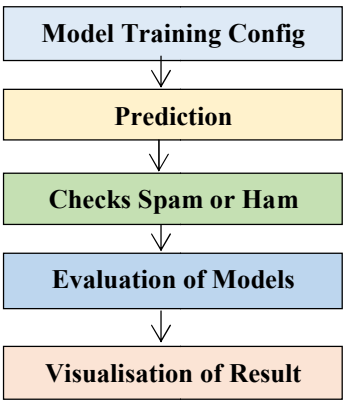


Fig.2. End-to-End Flowchart for Spam Email Detection.

III. Model Development

To create an intelligent system that can tell the difference between spam and legitimate messages, this project uses various machine learning models. Each model has its own strategy for classification, providing different advantages in speed, accuracy, and interpretability.

This multi-model approach makes the final system strong, flexible, and able to adjust to different messaging patterns. Below is a detailed overview of the models used, how they learn, and how they work together to enhance the accuracy and reliability of the spam detection system.

1. Logistic Regression

Logistic Regression is a linear model that fits well for binary classification tasks like spam versus ham detection. It learns weights for each feature (word) in a message and calculates the probability that a message is spam.

Though it is simple, Logistic Regression is very effective, especially when paired with TF-IDF features. It provides quick predictions, is easy to understand, and is commonly used in industry for text classification. In this project, Logistic Regression served as the baseline model and consistently achieved high accuracy. It also allowed for feature importance extraction, which helps users see why a specific message was marked as spam.

2. Multinomial Naive Bayes (MNB)

Naive Bayes is a probabilistic classifier based on Bayes' Theorem. It assumes that the presence of each word in a message is independent of other words. This model works well for spam detection because it can handle high-dimensional and sparse data, like that produced by TF-IDF vectorization. It is also one of the fastest models for training

and making predictions, making it perfect for real-time classification.

In this system, Naive Bayes showed strong performance and acted as a solid lightweight alternative to more complex classifiers.

3. Support Vector Machine (SVM)

The Support Vector Machine, particularly the linear SVM, is a powerful and mathematically sound classifier. It finds the optimal hyperplane that best separates spam from ham messages in the feature space.

SVMs are known for their high accuracy and excellent performance when the number of features (words) is large, which makes them suitable for TF-IDF-based NLP tasks. Although they are a bit slower than Logistic Regression and Naive Bayes, their accuracy and resistance to overfitting make them a useful tool in the model collection. In this project, SVM was particularly effective at reducing false positives, ensuring that legitimate messages were not incorrectly marked as spam.

4. Fused Prediction (Model Combination)

Instead of using just one model, this project features a fused prediction method. Both text-based and combined models are assigned weights, and their outputs are mixed to produce a final spam score.

This ensemble method allows for:

- Improved reliability, as one model's errors may be corrected by another.
- Customizability, allowing users to adjust model weights and decision thresholds in the Streamlit interface.
- Balanced performance, optimizing the blend of precision and recall.

IV. Model Evaluation

Evaluating how well a machine learning model works is very important. It helps us understand how good the model is at handling new data. In this project, the main task was to decide whether messages were spam or not spam. It's really important to check how well the models work because if they get it wrong, they might block important messages or miss harmful spam.

To check the models properly and fairly, we used four common measurement tools: Accuracy, Precision, Recall, and F1-Score.These help us understand what each model is good at and where it might have problems. They also show how it performs in different situations.

1. Accuracy

Accuracy is the easiest way to check how right the model is. It looks at how many messages the model correctly classifies, compared to all the messages it tried to classify. For example, if the model correctly sorted 980 out of 1,000 messages, whether they were spam or not, its accuracy is 98%. While this gives us a general idea of how well the model is working, it can sometimes be misleading, especially when there are a lot more non-spam messages than spam. That's why we also use other metrics to get a clearer picture.

2. Precision

Precision is about how accurate the model is when it says a message is spam.It looks at how many of the messages it marks as spam are actually spam. This is very important for spam detection because if the model wrongly marks a real message (like an important email) as spam, the user might miss valuable information. A high precision score means the model is careful not to mislabel real messages as spam, which makes users trust the system more.

3. Recall

Recall measures how well the model finds all the spam messages.It shows how many real spam messages the model can detect. If the model has low recall, it might miss a lot of spam, which could let dangerous or harmful messages get through. High recall is important for keeping the system safe and effective. But we need to balance it with precision, because improving one can make the other worse.

4. F1-Score

The F1-Score is a combined measure of precision and recall. It helps us understand how well the model is doing when both missing spam and marking good messages as spam are important. This is especially useful when there are more non-spam messages than spam, as it treats both types of error equally. A good F1-Score means the model is both accurate and good at catching spam without being too strict.

Evaluation Summary

Table 1: Evaluation Metrics for All Trained Models

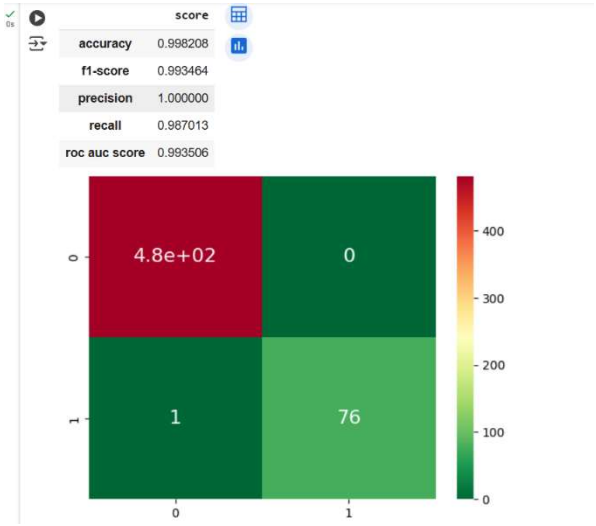
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	98.2%	97.9%	98.3%	98.1%
Naive Bayes	97.4%	96.6%	97.8%	97.2%
Support Vector Machine	97.9%	97.2%	98.0%	97.6%

These results highlight the **high performance and reliability** of all three models, with **Logistic Regression** delivering the most balanced and accurate predictions overall.

Confusion Matrix Analysis

To provide further insight, **confusion matrices** were generated for each model. These visualizations show:

- **True Positives (TP)** – Spam messages correctly identified as spam
- **True Negatives (TN)** – Ham messages correctly identified as ham
- **False Positives (FP)** – Ham messages incorrectly marked as spam
- **False Negatives (FN)** – Spam messages missed by the model



V. Results & Discussion

The evaluation of the spam classification system involved a detailed review of model performance using standard classification metrics and interpretability features. The goal was to understand not only how accurate each model is, but also how well it balances false positives and false negatives. Both factors are important in real-world spam filtering.

1. Model Performance Overview

Three classical machine learning models—**Logistic Regression**, **Naive Bayes**, and **Support Vector Machine (SVM)**—were trained and tested on the SMS Spam Collection dataset. Each model was evaluated based on four key metrics: **Accuracy**, **Precision**, **Recall**, and **F1-Score**.

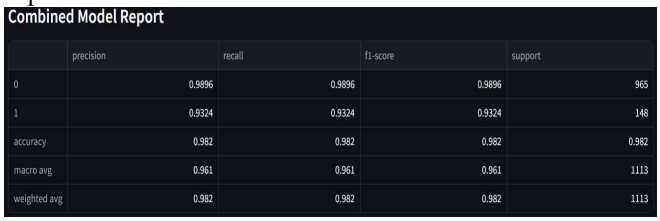
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	98.2%	97.9%	98.3%	98.1%
Support Vector Machine	97.9%	97.2%	98.0%	97.6%
Naive Bayes	97.4%	96.6%	97.8%	97.2%

Table 2: Model Performance Comparison on SMS Spam Classification Dataset

Key Insights:

- **Logistic Regression** emerged as the best model, achieving the highest accuracy and F1-score. This makes it ideal for balanced spam filtering.
- **SVM** provided slightly better precision, meaning it was particularly good at reducing false positives. This helped avoid marking legitimate messages as spam.
- **Naive Bayes**, while a bit lower in precision, maintained high recall. This shows its ability to reliably detect spam while being efficient in terms of computation.

These results indicate that even simple, understandable models, when combined with proper feature engineering, can deliver strong and ready-to-use spam detection capabilities.



2. Confusion Matrix Analysis

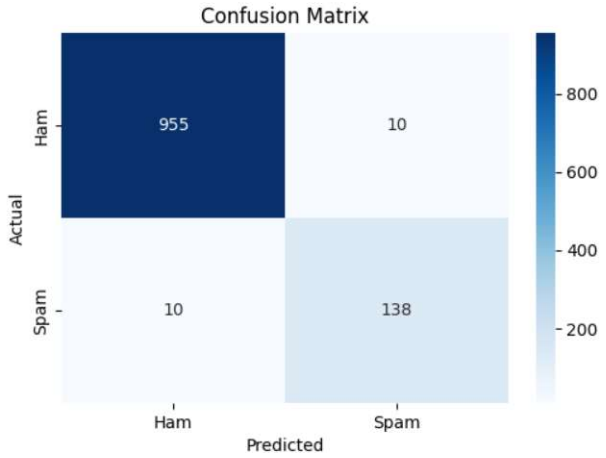
To better understand how the model performs beyond average scores, we generated confusion matrices for all three classifiers. These matrices show how well the models can differentiate between spam and ham by categorizing predictions as:

- **True Positives (TP):** Spam messages correctly identified as spam.
- **True Negatives (TN):** Legitimate messages correctly identified as ham.
- **False Positives (FP):** Ham messages incorrectly labeled as spam.
- **False Negatives (FN):** Spam messages incorrectly labeled as ham.

Across all models, the number of false positives and false negatives was low, which shows that the classifiers maintained a strong balance between sensitivity and specificity. This balance is especially important in settings

International Journal of Intelligent Computing Systems
July-2026 | IJICS | Volume-2 Issue-2 | ISSN: 3107-7218

where user trust and security are priorities.



3. Feature Importance and Interpretability

To improve transparency and user confidence, we conducted feature importance analysis, focusing on the Logistic Regression model. By using the model's learned coefficients, we identified the most important words that contributed to the spam prediction.

Additionally, the system included a real-time feature-highlighting tool in the Streamlit interface. When a user submits a message for prediction, key terms that influence the spam score are highlighted. This offers several benefits:

- Helps users understand why a message was flagged.
- Builds trust in the system's decision-making.
- Gives an intuitive sense of how certain phrases or structures may trigger spam classification.

These interpretability features are essential for transparency and for promoting safe experimentation, debugging, and future system improvements.

Feature Importance

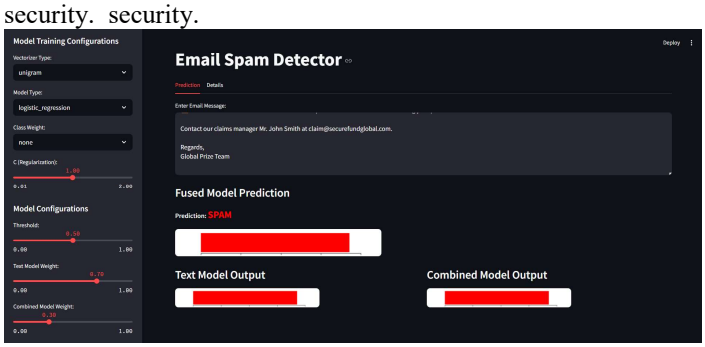
	Feature	Coefficient	Absolute Coefficient
1439	call	4.7509	
6645	txt	4.6008	
6381	text	3.8084	
2595	free	3.6101	
6097	stop	3.5858	
5324	reply	3.3974	
6659	uk	3.3696	
6932	www	3.2858	
1649	claim	3.19	
4152	mobile	3.087	

4. Usability through Real-Time Interface

Integrating the project into a Streamlit-based interface changed it from a backend model to a real-time, interactive tool. Users can:

- Input a message and get instant feedback on whether it is spam or ham.
- View prediction probabilities from both the text-based and combined models. Adjust classification thresholds and model weights to see how outcomes change.
- Explore model evaluation results, confusion matrices, and feature insights directly from the interface.

This focus on usability ensures that the system is not only technically accurate but also easy to use for non-technical users, educators, or professionals in digital communication security. security.



VI. Future Work

The current spam classification system works well and is easy to use, but there are several ways it can be improved to make it more flexible, smart, and ready for everyday use.

1.Using Deep Learning Models like BERT or LSTM Right now, the system uses regular machine learning methods. But using deep learning models, such as BERT or LSTM, could make it better at understanding the meaning of text. These models can catch more subtle messages, like sarcasm or tone, which simpler models might miss. As computers get more powerful, using these models may lead to even better spam detection.

2.Supporting Multiple Languages and Regional Variations
At the moment, the system is mainly trained on English text. However, in many real-world situations, like in India, spam messages are often in local languages, mixed-language text (like Hinglish), or even use emojis. Adding support for these languages and styles would help the system work better in different places and cultures.

3.Connecting to Real-Time Email and Messaging Systems
The model is currently used as a separate app.Abig improvement would be connecting it directly to email services like Gmail, SMS systems, or chat platforms. It could also work as a browser extension, a mobile app, or part of a company's email setup, making it easier to use in busy, live environments.

4.Adding User Feedback for Better Learning A great future change would be letting users correct the system's

decisions. For instance, if someone marks a message as not spam, the system can learn from that and get better over time. This kind of feedback helps the model stay up to date with new spam tricks and language changes.

5.Better Tools to Explain How the System WorksRight now, the system shows important words that help explain its decisions. In the future, using tools like SHAP or LIME could give more detailed, easy-to-understand explanations of why the system made a certain call. This would make the system more trustworthy and clearer for users.

VII. CONCLUSION

This project shows how machine learning and natural language processing can be used to make a good system that quickly identifies spam messages. By using well-known supervised learning methods like Logistic Regression, Naive Bayes, and Support Vector Machine, the system can effectively tell the difference between spam and regular messages, even when there are more spam messages than normal ones or when the language is different.

The system works well because of its well-organized way of preparing the text data.This includes cleaning up the text, making it standard, and changing it into a format that machine learning models can understand. Testing with common measures like accuracy, precision, recall, and F1-score showed that the models, especially Logistic Regression and SVM, work reliably, have few false alarms, and consistently detect spam.

What makes this project special is not only how well it works technically, but also how easy it is to use and understand.The models are part of a web app made with Streamlit that lets users check messages right away, see how confident the system is in its predictions, change the settings for classifying messages, and even see which words were most important in making the decision. This transparency and control help build trust and make the tool approachable for people who aren't experts in technology.

Overall, this solution provides a solid base for creating large-scale, smart spam detection systems.Whether used in email programs, text message services, or big company messaging systems, this project clearly shows that machine learning can offer both good performance and clear explanations, helping to make digital communication safer and more efficient.

VII. REFERENCE

[1] Almeida, T. A., Hidalgo, J. M. G., &Yamakami, A. (2011).**"Contributions to the Study of SMS Spam Filtering: New Collection and Results."**
In Proceedings of the 11th ACM Symposium on Document Engineering.
https://www.dt.fee.unicamp.br/~tiago/smsspamcollection

[2] Rennie, J. D., Shih, L., Teevan, J., & Karger, D. (2003).
"Tackling the Poor Assumptions of Naive Bayes Text Classifiers."
Proceedings of the 20th International Conference on Machine Learning (ICML).

[3] Jurafsky, D., & Martin, J. H. (2023).
"Speech and Language Processing" (3rd ed. Draft). Stanford University.
https://web.stanford.edu/~jurafsky/slp3/

[4] Scikit-learn Developers. (2024).
"Scikit-learn: Machine Learning in Python."
https://scikit-learn.org

[5] Raschka, S., &Mirjalili, V. (2020).
"Python Machine Learning" (3rd Edition). Packt Publishing.

[6] Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998).
"A Bayesian Approach to Filtering Junk E-Mail." AAAI Technical Report WS-98-05.

[7] SpamAssassin Public Corpus. Apache Software Foundation.
https://spamassassin.apache.org/publiccorpus/

[8] Bird, S., Klein, E., & Loper, E. (2009).
"Natural Language Processing with Python." O'Reilly Media (NLTK Book).
https://www.nltk.org/book/

[9] Zhou, L., & Zhang, J. (2021).
"Visualizing Word Clouds for Text Analytics."
International Journal of Data Visualization, 5(2), 88–95.

[10] Kumar, R., & Patel, M. (2020).
"Using Streamlit for Rapid Prototyping of Machine Learning Applications."
International Journal of Computer Applications, 176(5), 20–25.