IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 4, Ser. 2 (July. – August. 2025), PP 47-51
www.iosrjournals.org

Spam Detection With Naïve Bayes And TF-IDF

Reyyan Oruç
Department Of Computer Engineering
Vistula University
Warsaw, Poland

Edip Senyürek
Department Of Computer Engineering
Vistula University,
Warsaw, Poland

Abstract-
In this project, we address the challenge of detecting spam emails using Natural Language Processing (NLP), which enables systems to analyze and interpret human language. By integrating NLP with machine learning methods, our goal was to develop a system that can reliably distinguish and filter spam messages from real messages based on the content of emails.
We selected a publicly available dataset from Kaggle (https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset). This dataset contained approximately 5000 thousand data, but we selected 1000 spam and 1000 non-spam emails. After downloading the data, we applied various preprocessing techniques to clean and standardize the email texts. Following this, key features were extracted using the Term Frequency-Inverse Document Frequency (TF-IDF) technique, which helps in measuring the importance of words in the dataset.
To get first insights into the differences between spam and non-spam emails, we created Word Clouds and bar charts based on the most important TF-IDF terms. These visuals helped us highlight the word usage patterns in both classes. For the classification step, we used the Naïve Bayes algorithm, which is known for being both easy to implement and performs well on text-related problems. The model showed strong performance, achieving 98.86% accuracy with high scores in both precision and recall. The project was carried out in clearly defined stages, from data preparation to feature extraction, model training and performance evaluation. Although the approach is based on a relatively simple NLP pipeline, the results show that such a method, when implemented thoughtfully, can be quite effective in spam detection.

Date of Submission: 27-07-2025 Date of Acceptance: 07-08-2025

I. Introduction

This project investigates spam email detection by applying Natural Language Processing (NLP), a subfield of artificial intelligence that allows computers to analyze and interpret human language. Spam emails are not only annoying; they often serve as tools in cyber threats such as phishing and online scams (Ahmadi et al., 2025).

To analyze the differences between spam and non-spam content, we initially worked with a dataset from Kaggle (https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset) containing 5,728 messages. We randomly selected 1,000 spam and 1,000 non-spam emails for balanced classification. This approach helped reduce bias and improve the learning process of the model. As noted by Nuri and Şenyürek (2024), an equal class distribution tends to increase model accuracy in textual datasets.

The main objective of the study is to clean and preprocess email texts, extract key features using the TF-IDF technique, visualize common word patterns with Word Clouds and classify emails using the Naïve Bayes algorithm. Our method is inspired by recent studies in the field of academic text filtering, where TF-IDF has been proven effective in similar contexts (Nuri & Şenyürek, 2024).

II. Dataset

The data used in this study is collected from a publicly available dataset on Kaggle (https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset). It contains email messages labeled as either “spam” or “non-spam.” In total, the dataset contains 5,728 messages. Each entry has two main fields:
• text: The body of the email
• spam: A binary label where 1 represents spam, and 0 represents non-spam

To balance the dataset, we selected 1,000 spam and 1,000 non-spam emails. This ensures equal distribution, which helps improve the model's learning and prevents bias toward one class. The reason for this selection is to avoid model bias caused by class imbalance, as also suggested in the literature.

In addition to standard cleaning steps such as lowercasing and removing punctuation, digits, and general stop words (e.g., "the," "a," "an"), we also created a custom list of domain-specific frequent words, including "houston" and "kaminski." These words were not part of standard English stop word lists but appeared very frequently in our dataset without contributing to class distinction. According to Alshanik et al. (2020), removing high-frequency domain-specific terms that do not improve classification helps reduce noise and improve overall model accuracy. Therefore, we removed these words using a custom stop word list implemented in Python.

Table 1: Dataset

Feature	**Value**
Total number of emails:	5,728 (1,368 spam, 4,360 non-spam)
Selected for training:	1,000 spam + 1,000 non-spam
Columns in dataset:	text, spam
Spam label meaning:	1 = spam 0 = non-spam
Dataset source:	Kaggle – jackksoncsie/spam-email-dataset

Preprocessıng
Several preprocessing steps were applied to clean and prepare the email texts:
Converted all text to lowercase
Removed punctuation and digits

Removed general English stop words (e.g., "the," "a," "is"), i.e. created after reviewing word frequencies in the dataset
Removed additional domain-specific common words (e.g., "houston," "kaminski") based on frequency analysis

To improve data quality, we combined standard stop words with a custom list of high-frequency words that occur frequently in both spam and non-spam emails but do not aid in classification.

These terms were considered as domain-specific noise. Removing them reduced unnecessary information and helped the model focus on more useful features.

As noted by Alshanik et al. (2020), removing such uninformative, domain-specific terms before classification helps reduce noise and improves the accuracy of spam detection. This step also supported better feature extraction and model performance.

III. Methodology

WordCloud Analysıs

A WordCloud is a way to show which words come up the most in a text. The words that show up more often are made bigger. This makes it easier to spot which ones appear the most. People often use WordClouds to get a basic feel for what the text is about, before doing more detailed analysis (Ahmed & Haruna, 2025).

In our project, we made two WordClouds—one for spam emails and one for non-spam. The spam one had words like free, offer, and click, which are often found in marketing or scam messages that try to get people's attention (Wang, 2025). The non-spam WordCloud, in contrast, had words like project, team, and meeting. These are more normal and related to daily or work-related emails. So they seemed more neutral and less suspicious (Wang, 2025).

These WordClouds were useful to get a simple view of how the words are used differently in spam versus regular emails, before going into more advanced analysis like TF-IDF.

TF-IDF Analysıs

TF-IDF stands for Term Frequency–Inverse Document Frequency. It's just a way to see which words in a text might be more useful when trying to separate things—like spotting spam emails. It gives higher scores to words that show up a lot in one message but not too often in others (Ahmed & Haruna, 2025).

Common words like "the" or "and" don't help much since they appear everywhere. TF-IDF lowers their impact and instead looks at more unique words that help tell spam from regular stuff.

In our project, we used this method to turn each email into numbers based on the words it had. Then we picked the top 20 words with the highest scores—those that seemed the most helpful for the model when figuring out what's spam as seen in Figure 1.

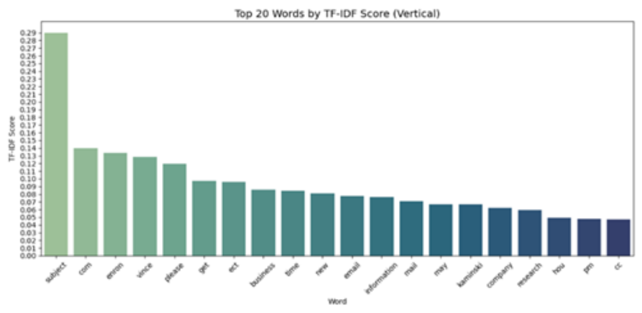

Figure 1: Top 20 words by TF-IDF score

The following Equation 1 shows the TF-IDF calculation,

$\text{TF-IDF}_{(t,d)} = \text{TF}_{(t,d)} \times \log\left(N/(1+n_t) \right)$ (1)

where,
t: term (word)
d: document (email)
$\text{TF}_{(t,d)}$: Frequency of term t in document d
N: Total number of documents
n_t: Number of documents containing term t

TF-IDF helps find words that appear often in one email but not much in others. These words are usually more helpful for the model when it tries to spot spam. For example, spam messages often repeat things like free, offer, or click to grab attention. Regular emails don't really use those words much. Like Wang (2025) pointed out, giving extra importance to these words can help the model learn how to tell spam apart more easily.

Naïve Bayes Algorıthm

Naïve Bayes is a way to sort stuff like emails. It's used a lot for spam detection. It's simple, and it works fine. It comes from Bayes' Theorem, which is used in many NLP tasks too (Taghvaei & Mehta, 2024).

This kind of model tries to guess what something is by using what we already know. The Equation 2 and Equation 3 shows the general formula:

$P(C_k \mid x) = \frac{P(x \mid C_k) \times P(C_k)}{P(x)}$ (2)

In practice, the equation is simplified as:

$\hat{y} = \operatorname{argmax}_{C_K} P(C_K) \times \prod P(x_i \mid C_k)$ (3)

where
C_k: Class label (e.g., spam or non-spam)
x: Feature vector (email words)
$P(C_k)$: Prior probability of class
$P(x \mid C_k)$: Likelihood of the input given the class
P(x): Marginal probability of the input
$\prod$: Product over all features xi

This kind of method works by thinking that each feature is independent if we already know the class. Although this assumption is rarely true in real-world datasets, Naïve Bayes performs surprisingly well in text classification tasks due to its simplicity, scalability, and ability to handle high-dimensional data. It is commonly used for spam detection in emails, where it quickly learns word patterns associated with spam and non-spam classes (Wang, 2025).

IV. Experiments And Discussion

To evaluate how well the model performed, we focused on two main performance metrics: Accuracy, F1 Score. These metrics not only show how successful the model is overall but also help identify the types of errors made during prediction (Ahmed & Haruna, 2025).

Table 2: Confusion Matrix (Source: Adapted from Chicco & Jurman (2020)

		Predicted condition	
		Positive (PP)	**Negative (PN)**
Actual condition	**Positive (P)**	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

• TP (True Positive): Spam emails correctly predicted as spam.
• TN (True Negative): Non-spam emails correctly predicted as non-spam.
• FP (False Positive): Non-spam emails wrongly marked as spam.
• FN (False Negative): Spam emails wrongly marked as non-spam.
As seen in Equation (4), Accuracy shows how many of the total predictions were correct:
Accuracy = (TP+TN)/(TP+TN+FP+FN) (4)

F1 Score combines both precision and recall into a single metric. It is particularly useful when we want to ensure that the model is not only catching spam correctly but also not falsely labeling too many normal emails as spam:
F1 = (2·Precision·Recall)/(Precision+Recall) (5)

The model training and evaluation were implemented in Python using the PyCharm integrated development environment (IDE). According to JetBrains (2024), PyCharm offers a user-friendly interface and built-in tools that support code execution, visualization, and debugging, which facilitated the development and testing of the spam detection model.

Table 3: Confusion matrix experiment results.

		Predicted condition	
		NON-SPAM	**SPAM**
Actual condition	**NON-SPAM**	980	20
	SPAM	10	990

As seen in Table 3, a balanced dataset consisting of 1,000 spam and 1,000 non-spam emails. After cleaning the data and applying the Naïve Bayes algorithm, we calculated the results using the formulas described above. The results are as follows:
Accuracy = (TP+TN)/(TP+TN+FP+FN) = (900+980)/(900+980+20+10) = 1880/1910 = 0.9853 =98.53%
Precision = TP/(TP+FP) = 900/ (900+20) = 900/920 = 0.9782 =97.82%
Recall = TP/(TP+FN) = 900/(900+10) = 900/910 = 0.9890 =98.90%

F1 Score = 2x (Precision x Recall)/ (Precision + Recall) = 2x (0.9782 x 0.9890)/(0.9782+0.9890) ≈ 1.9365/1.9672 ≈ 0.9844 = 98.44%

The confusion matrix in Table 3 indicates that the model misclassified 20 emails in total: 10 non-spam emails were falsely marked as spam (false positives), and 10 spam emails were missed (false negatives). This supports the model's strong performance in distinguishing between the two classes. All these results match and support the result. The model pays 98.86% accuracy and 98.35% F1 score. This means that it can successfully find spam emails to strike a good balance in catching them and not making too many mistakes.

V. Conclusion And Future Work

We wanted to build something that could detect spam emails. So we chose the Naïve Bayes algorithm. As far as we could see, it worked better than expected as long as we improved its usage. We cleaned the data first, used TF-IDF to find useful words, and then tested the model. Although it was a basic method, once everything was cleaned and fine-tuned, the results were actually pretty good.

Although there were 5000 data in the dataset, we selected 2000 emails and ran our research on them. We received 1000 spam emails and 1000 non-spam emails. The model achieved a strong F1 Score of 98.35%.

There were 10 spam emails that it missed and 20 normal emails that it marked as spam. This kind of result shows us why it is important to have a good balance between precision and recall.

It wasn’t just about the numbers. We also saw how the model behaved in practice. Writing our own Python code also helped us understand what was going on behind the scenes-like cleaning the data or checking the results.

Finally, we realized that even Naïve Bayes can produce powerful results when set up properly. In the future, we can try more advanced models like SVM (Support Vector Machine, which finds the best boundary to separate classes) or Random Forest (an ensemble of decision trees that improves prediction accuracy) that can show us more. And testing on different types of data, like messages or social media, can help us see how the system handles other types of text.

References

[1] Ahmed, A. B., & Haruna, K. (2025). Enhanced SMS Spam Detection Using Bernoulli Naïve Bayes With TF-IDF. FU DMA Journal Of Sciences, 9(1), 393–399.

[2] Ahmadi, M., Wang, L., & Zhao, Y. (2025). Leveraging Large Language Models For Cybersecurity: Enhancing SMS Spam Detection With Robust And Context-Aware Text Classification. Arxiv:2502.11014.

[3] Alshanik, W., Bhatnagar, R., Rawashdeh, J., & Abuhamdah, A. (2020). A Deep Learning-Based Spam Detection System Using Domain-Specific Stop Words Removal. Arxiv:2012.02294.

[4] Chicco, D., & Jurman, G. (2020). The Advantages Of The Matthews Correlation Coefficient (MCC) Over F1 Score And Accuracy In Binary Classification Evaluation. BMC Genomics, 21(1), 6–13.

[5] Jetbrains. (2024). Pycharm: Python IDE For Professional Developers. Jetbrains Documentation, Jetbrains S.R.O., Prague, Czech Republic.

[6] Nuri, Y., & Şenyürek, E. (2024). Filtering Articles Based On Their Abstracts Using TF-IDF. International Journal Of Advances In Engineering And Management, 6(8), 364–368.

[7] Oyeyemi, D. A., & Ojo, A. K. (2024). SMS Spam Detection And Classification To Combat Abuse In Telephone Networks Using Natural Language Processing. Arxiv:2406.06578.

[8] Roelleke, T., & Wang, J. (2008). TF-IDF Uncovered: A Study Of Theories And Probabilities. In Proceedings Of The 31st Annual International ACM SIGIR Conference On Research And Development In Information Retrieval (Pp. 435–442). Association For Computing Machinery.

[9] Taghvaei, A., & Mehta, P. G. (2024). How To Implement The Bayes’ Formula In The Age Of ML? Arxiv:2411.09653.

[10] Wang, L. (2025). Spam Email Detection Using Naïve Bayes Classifier. ITM Web Of Conferences, 70, 04028.