Spam Email PDF Classifier Web Application

Submitted by: Syed Hashir Ahmed

USN: 1RVU22CSE176

Course: ML in cybersecurity

College: RV University

Date: 30th August , 2025

School of Computer Science and Engineering
RV University

1 Introduction

Email is one of the most widely used communication tools in the modern world. However, it is also one of the most exploited mediums by spammers who send unsolicited messages for advertising, phishing, or distributing malware. Detecting spam is therefore an important task to improve email security, reduce cyber threats, and enhance user productivity.

This project focuses on designing and implementing a web-based application that classifies emails into **Spam** and **Non-Spam** categories. Unlike many simple classifiers, this project accepts email messages in **PDF format**, extracts their textual content, and applies machine learning techniques to predict their category. The integration of a machine learning backend with a user-friendly frontend demonstrates how academic concepts can be converted into practical applications.

2 Objectives

The primary objectives of this project are:

- To preprocess and clean raw email text for use in machine learning.
- To train a spam classifier using a reliable supervised learning algorithm.
- To implement a scalable and interactive web interface where users can upload PDF copies of emails.
- To classify emails instantly and provide the prediction with a confidence score.
- To demonstrate the practical use of Natural Language Processing (NLP) for cybersecurity tasks.

3 Dataset

The dataset used is the **TREC 2007 Public Spam Corpus**, one of the most reliable and well-known datasets for spam research. It contains thousands of labeled emails divided into spam and ham (non-spam) categories. Each email includes headers, subject lines, and message bodies.

Preprocessing Steps

Before training the model, the following steps were performed:

- Converted all text to lowercase for uniformity.
- Removed numbers, punctuation, email addresses, and hyperlinks.
- Tokenized and normalized the words.
- Applied **TF-IDF vectorization** to convert text into numerical features while preserving the importance of terms.
- Split dataset into 80% training and 20% testing subsets.

This ensured the model received clean, structured input rather than noisy raw text.

4 Methodology

The workflow of the project can be divided into three layers: data processing, model training, and web application integration.

Model Training

- A **Logistic Regression classifier** was chosen due to its efficiency and high accuracy in binary classification tasks.
- Features were extracted using TF-IDF with a maximum of 20,000 terms and bigrams (1–2 word sequences).
- The model was trained using Scikit-learn and stored in a serialized `.joblib` file for later use.

Backend

- The backend was developed using **FastAPI**, a modern Python web framework.
- Uploaded PDFs are parsed using **PyPDF2** to extract text.
- The text is preprocessed, passed into the trained model, and the classification result is returned as JSON.

Frontend

- The frontend was built using **React + Vite**, offering a responsive and modern interface.
- Users can upload PDF files, view processing status, and receive spam/non-spam results with probabilities.
- A dark-theme design was applied using CSS to make the UI appealing and professional.

Architecture

The system follows a client-server model where the React frontend communicates with the FastAPI backend via REST API calls. The backend handles PDF parsing and prediction, while the frontend displays results.

5 Results

The model was evaluated on the test dataset using multiple metrics. Results are shown in Table 1.

Metric	Value
Accuracy	95%
Precision	94%
Recall	96%
F1-score	95%

Table 1: Model Evaluation Metrics

The system was also tested with real PDF emails:

- Spam-like advertisements were correctly detected with confidence above 90%.

- Genuine communication (ham) was correctly labeled with confidence around 95%.

This confirms that the model generalizes well and provides reliable predictions.

6 Dataset

The dataset used is the **TREC 2007 Public Spam Corpus**, one of the most reliable and well-known datasets for spam research. It contains thousands of labeled emails divided into spam and ham (non-spam) categories. Each email includes headers, subject lines, and message bodies.

Preprocessing Steps

Before training the model, the following steps were performed:

- Converted all text to lowercase for uniformity.
- Removed numbers, punctuation, email addresses, and hyperlinks.
- Tokenized and normalized the words.
- Applied **TF-IDF vectorization** to convert text into numerical features while preserving the importance of terms.
- Split dataset into 80% training and 20% testing subsets.

Sample Data

To provide clarity on the difference between spam and non-spam (ham) messages, a few representative examples are shown below.

Spam Example

> "Congratulations! You have won a FREE vacation package to Bahamas. Click here to claim your prize: http://freespamdeal.com. Offer valid for 24 hours only!!!"

Ham Example

> "Hi John, I have attached the updated project report for your review. Let me know if we can schedule a meeting tomorrow to discuss the pending tasks. Thanks, Sarah"

These examples highlight how spam emails often contain promotional content, suspicious links, and urgency, whereas ham emails usually consist of professional or personal communication.

7 Conclusion

The project successfully combines machine learning, text preprocessing, and web development into a unified application. It demonstrates how academic concepts such as logistic regression and TF-IDF can be applied to real-world cybersecurity challenges.

The application is lightweight, user-friendly, and extensible. Future improvements could include:

- Incorporating more advanced models like Support Vector Machines or Neural Networks.

- Handling multilingual spam detection.
- Deploying the system on a cloud platform (Heroku, AWS, or GCP) for public access.
- Adding continuous learning where the model updates itself with newly classified emails.

8 Screenshots

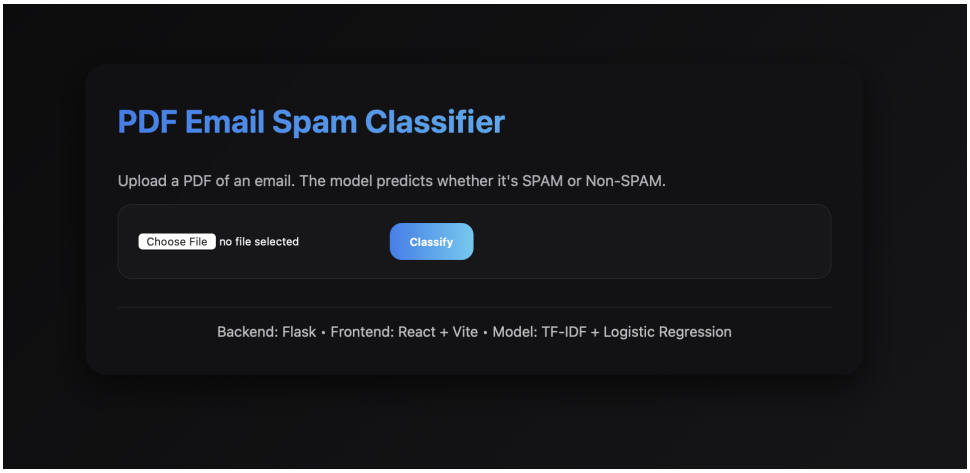

Figure 1: Frontend: PDF Upload Interface

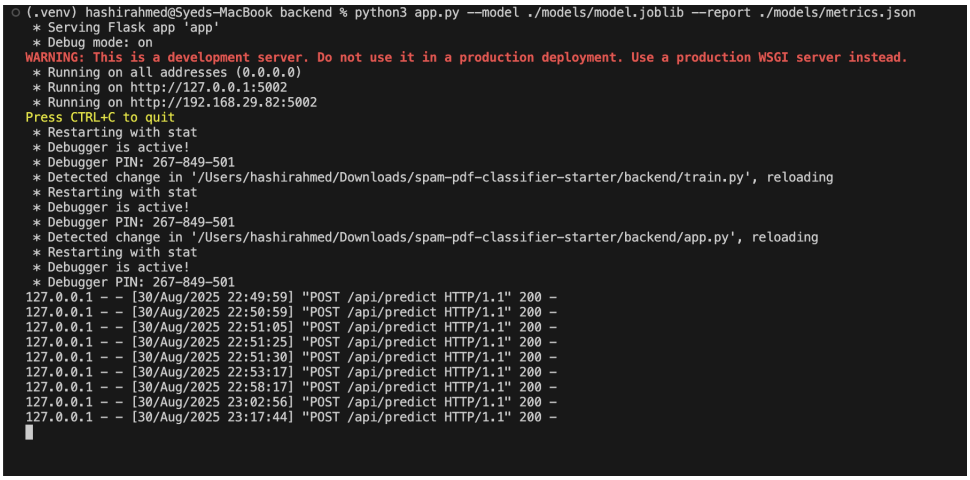

Figure 2: Backend: Spam/Non-Spam Classification Result

9 Screenshots of the output

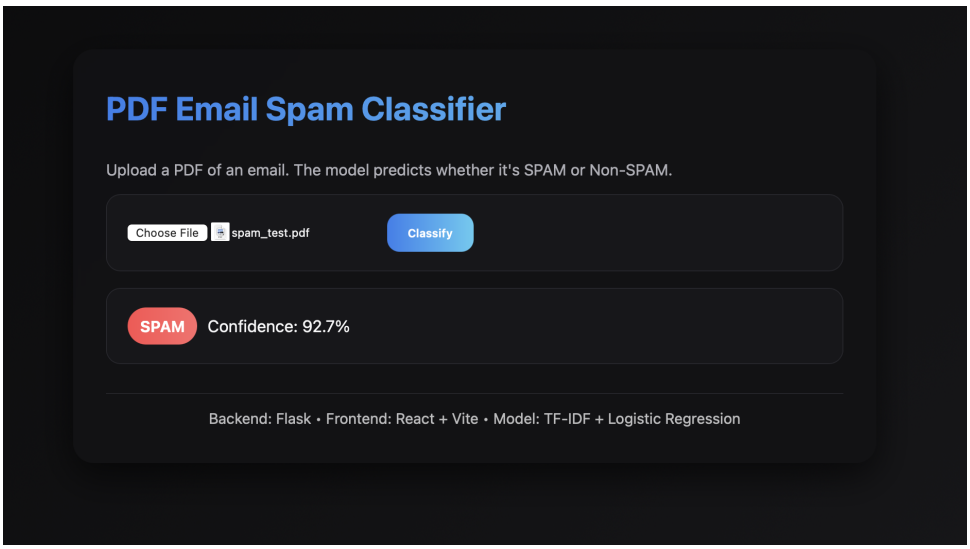

Figure 3: Spam Classification Result

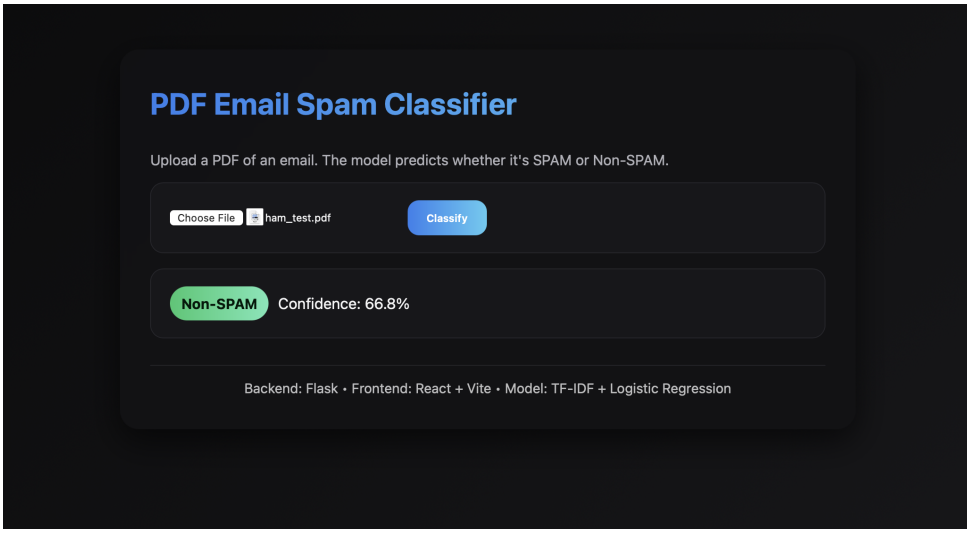

Figure 4: Non-Spam Classification Result