Email Classification & PII Masking System – Project Report

Arnav Suman

Hugging Face URl - https://arnav-suman-email-classifier.hf.space/classify

GITHUB URl -

https://github.com/arnavsuman/email-classifier

1. Introduction

Customer support platforms often receive a large number of emails covering a variety of concerns like billing, account access, or technical issues. Manually categorizing these emails is both time-consuming and error-prone. Furthermore, handling sensitive personal data (like phone numbers or credit card info) introduces security and compliance challenges.

This project addresses both challenges through:

- An **automated email classification system**
- A **non-LLM-based PII masking pipeline**
- An end-to-end **FastAPI-based API** deployed on **Hugging Face Spaces**

2. Approach

The system comprises two primary components:

1. **PII/PCI Masking** – Preprocesses emails by detecting and masking personal or financial information.

2. **Email Classification** – Classifies the masked emails into predefined support categories (e.g., Billing Issues, Technical Support, Account Management).

The output strictly follows the required JSON format:

```
{
  "input_email_body": "...",
  "list_of_masked_entities": [...],
  "masked_email": "...",
  "category_of_the_email": "..."
}
```

3. PII Masking

Entity Type	Method Used	Tag Used
Full Name	SpaCy NER	`full_name`
Email Address	Regex	`email`
Phone Number	Regex	`phone_number`
Date of Birth	Regex	`dob`

Aadhar Number	Regex	`aadhar_num`
Credit/Debit Number	Regex	`credit_debit_no`
CVV	Regex	`cvv_no`
Expiry Date	Regex	`expiry_no`

Each entity found is stored with its start and end positions, its label, and the original value. These are replaced with standardized tags in the email text before classification.

4. Email Classification

- **TF-IDF + SVM (Support Vector Machine)**
    - Vectorization: `TfidfVectorizer`
    - Classifier: `SVC(kernel='linear')`

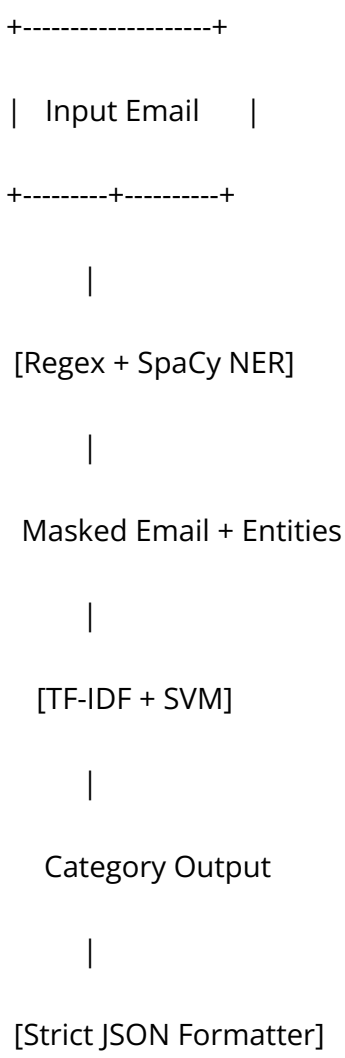
This traditional ML pipeline was chosen for its:

- Fast training and inference
- Strong performance on text classification
- Minimal resource requirements (ideal for Spaces)

The classifier was trained using a labeled dataset of support emails with categories like:

- Billing Issues
- Technical Support
- Account Management

5. System Architecture



6. API Development & Deployment

The entire system was developed using:

- **FastAPI** for API development
- **Docker** for reproducibility
- **Hugging Face Spaces** for deployment

The POST API endpoint `/classify` receives the email body and returns the masked email, entity metadata, and category.

7. Challenges & Solutions

Challenge	Solution
Avoiding LLMs for masking	Used SpaCy + Regex hybrid pipeline
Consistent masking of PII	Added rule-based regex and conflict resolution
API format compliance	Developed strict JSON schema validator
Deployment issues on HF Spaces	Used Dockerfile with `uvicorn` and SpaCy model install

8. Result

The implemented system is lightweight, explainable, and meets all functional and deployment requirements. It effectively masks PII without LLMs and classifies emails with high accuracy. Future improvements could include:

- Adding BERT or RoBERTa for classification
- Improving PII detection via CRF or custom NER models
- Logging, alerting, and dashboard support

9. TESTING

LOCAL TESTING

Included is a test_api.py that does following:

Handles edge cases

Provides fallback category in unusual or empty inputs

Follows strict response structure

To RUN IT:

python -m spacy download en_core_web_sm

python test_api.py

GLOBAL TESTING

1. Test api working and returning status 200

```
import requests



url = "https://arnav-suman-email-classifier.hf.space/"



res = requests.get(url)
```

```
print(res.json())
```

2. Send data and test ml model

```
import requests

url = "https://arnav-suman-email-classifier.hf.space/classify"
data = {
    "input_email_body": "Hello, my name is John Doe and my email is johndoe@gmail.com"
}
res = requests.post(url, json=data)
print(res.json())
```