Email Classification and PII Masking System

🔹 Introduction

In modern organizations, support teams receive a large volume of emails daily, covering a wide range of issues such as billing queries, account management requests, and technical problems. Efficiently classifying these emails helps streamline operations and ensure timely responses.

However, many support emails contain Personally Identifiable Information (PII) such as names, phone numbers, emails, and financial details. To comply with data privacy regulations and secure customer data, it is critical to detect and mask this sensitive information before processing.

This project aims to develop a full pipeline that:

- Detects and masks PII/PCI information from support emails.
- Classifies the masked emails into predefined categories.
- Returns the classification result and a demasked email in a secure manner.
- Exposes this solution through an API deployed on Hugging Face Spaces using FastAPI

🔹 Approach Taken

The pipeline designed for this project consists of three primary components:

1. PII Detection and Masking (Non-LLM Approach)

PII masking is done **without using Large Language Models**, as per the problem requirements. Instead, the solution uses **Regex-based Named Entity Recognition (NER)** for extracting and masking fields such as:

- full_name
- email
- phone_number
- dob
- aadhar_num
- credit_debit_no
- cvv_no

- expiry_no

The masking replaces sensitive data with a label, for example:

text

Copy code

Original: My email is johndoe@gmail.com.

Masked: My email is [email].

2. Email Classification

A rule-based classifier was implemented using simple keyword detection logic due to time constraints and lack of labeled training data. The model currently checks for category-specific keywords like:

- "invoice", "payment" ⟶ *Billing Issues*
- "login", "error" ⟶ *Technical Support*
- "account" ⟶ *Account Management*
- Fallback ⟶ *Request*

This can later be improved by training traditional ML models or transformer-based classifiers if a labeled dataset is provided.

3. API Development and Deployment

The final system was developed using **FastAPI** and exposed via a /classify POST endpoint. The system returns:

- input_email_body (original content)
- list_of_masked_entities (detected PII with type, span, and original value)
- masked_email (with placeholders)
- category_of_the_email (classification label)

This API was successfully deployed on Hugging Face Spaces at:

arduino

Copy code

https://Harshita24-email-classifier.hf.space/classify

🔹 Model Selection & Training Details

Due to constraints in time (1 hour) and lack of labeled data, the classifier is **rule-based**. However, this is modular and can be replaced with:

- **Naïve Bayes or SVM** (for small datasets)
- **Fine-tuned BERT/RoBERTa** (for large datasets with labeled categories)
- **Custom LSTM/CNN Text Classifiers** (for sequence learning tasks)

No separate training was performed, but the code is structured to allow easy integration of trained models using scikit-learn, transformers, or TensorFlow.

🔹 Challenges Faced

Challenge	Solution
Time constraint for development	Designed modular, minimal, working code with regex + rule-based classifier
Regex limitations for PII masking	Carefully tuned regular expressions to reduce false positives
FastAPI + Hugging Face config error	Fixed by correctly setting up app.py, requirements.txt, and correct Space settings
Model selection	Opted for a rule-based solution as a working MVP, easily replaceable by ML/DL

🔹 Conclusion

The developed system fulfills all the assignment requirements:

- Detects and masks PII using non-LLM methods
- Classifies emails into predefined support categories
- Exposes a working API at the expected /classify router
- Correctly follows input-output structure and JSON format

The system is lightweight, extendable, and deployable in real-world support settings, and can be enhanced in the future with better classification models and entity extraction techniques.