Email Classification with PII Masking API

Introduction

With the increasing use of emails in customer support and internal communication, organizations often receive emails that contain personally identifiable information (PII) such as names, phone numbers, card details, and emails. At the same time, these emails need to be categorized efficiently to streamline service processes.

This project solves both problems by:

1. Automatically detecting and masking sensitive PII using regular expressions.
2. Classifying the emails into one of four categories: Incident, Request, Change, or Problem using a machine learning model.

The solution is implemented as a FastAPI application with a single /classify endpoint, designed for easy testing and deployment, including on platforms like Hugging Face Spaces.

PII Masking

Goal: Replace sensitive parts of an email with standard labels (e.g., [email], [phone_number]) to protect privacy before classification.

Method:
Used regular expressions (regex) to detect PII in the email text.

Defined specific regex patterns for each PII type:

- Full Name
- Email Address
- Phone Number
- Date of Birth
- Aadhar Number
- Credit/Debit Card Number
- CVV
- Expiry Date

Each match is masked and logged with position, label, and original value.

Overlap Handling:

- Implemented logic to avoid overlapping matches, so only one label applies per section of the email text.

Email Classification

- **Goal:** Identify the category of the email (Incident, Request, Change, or Problem).
- **Dataset:** A CSV file (combined_emails_with_natural_pii.csv) with real-looking emails and categories.
- **Preprocessing:**
  - Removed PII using the masker.
  - Transformed email text using TF-IDF vectorization.
- **Model:** Multinomial Naive Bayes classifier (scikit-learn) chosen for speed, simplicity, and high accuracy.
- **Training Output:** Model and vectorizer saved in model.pkl using pickle.

Main Files

- **pii_masker.py:** Contains regex patterns and masking logic.
- **classifier.py:** Trains and predicts email categories.
- **app.py:** FastAPI application exposing a /classify POST endpoint.

API Endpoint

URL: /classify
Method: POST

Input:

```
{
 "input_email_body": "Hi, my name is John Doe and my card number is 1234 5678 9876 5432"
}
```

Output:

```
{
 "input_email_body": "...",
 "list_of_masked_entities": [
  {
   "position": [18, 26],
   "classification": "full_name",
```

```
    "entity": "John Doe"
   },
   ...
 ],
 "masked_email": "Hi, my name is [full_name] and my card number is
[credit_debit_no]",
 "category_of_the_email": "Request"
}
```


Dependencies

fastapi
uvicorn
pandas
scikit-learn
pydantic

Challenges and Solutions

Challenge	Solution
Overlapping regex matches	Implemented a check to skip overlapping spans
Regex false positives	Refined patterns and added context
Maintaining strict output format	Used Pydantic models and controlled API responses
Model performance on short emails	Included enough data and consistent preprocessing
Deployment on Hugging Face	Tested locally, structured repo properly

Conclusion

This project demonstrates how to create a privacy-preserving email classification system using classic machine learning and regex. It meets all assignment requirements, provides a clean and reusable API interface, and is ready to be deployed or extended for more complex NLP tasks.