Email Classification System with PII Masking

Final Report - Akaike Technologies Assignment

1. Objective

The primary objective of this project is to build a secure and reliable pipeline that classifies incoming support emails into relevant categories while detecting and masking Personally Identifiable Information (PII). The system ensures that no sensitive information is exposed during processing, and only the masked version is used for classification. The pipeline exposes a FastAPI-based REST API endpoint that performs masking, classification, and returns results in a strict JSON format.

2. Project Overview

- Input: An email body submitted via an API POST request
- Output:
  - Original input email
  - List of detected PII entities with position, type, and original value
  - Masked email with PII replaced by placeholder tokens
  - Predicted category of the email (Request, Incident, Problem, or Change)

The entire flow is containerized and deployable using Docker on Hugging Face Spaces.

3. Approach

a. PII Masking

- Regular expressions are used to detect structured PII types:
  - Email addresses
  - Phone numbers
  - Credit/Debit card numbers
  - Aadhar numbers
  - CVVs and expiry dates
- spaCy's English NER model (en_core_web_sm) is used to detect PERSON names
- Detected entities are replaced with placeholder tokens like [email], [full_name], etc.
- Position offsets and entity metadata are captured in the output

b. Email Classification

- A TF-IDF + Naive Bayes pipeline is built using scikit-learn
- Input features: masked email text
- Target label: type column in the dataset (Request, Incident, etc.)
- Training data: provided CSV file containing emails and categories
- Classification report shows acceptable performance across all classes

4. FastAPI Integration

- Built with FastAPI and exposed as a RESTful POST endpoint at "/"
- Accepts a JSON input: { "input_email_body": "..." }
- Returns structured output JSON with masked entities and predicted label
- Uses Swagger UI at /docs for interactive testing
- Automatically opens browser to /docs when launched via python app.py

5. Technologies Used

- Python 3.10
- FastAPI
- scikit-learn
- spaCy
- pandas, numpy
- Regex for pattern matching
- Docker for deployment
- Hugging Face Spaces for public API hosting

6. Sample Input & Output

Input:

```
{
    "input_email_body": "Hey, I'm Raj. My email is raj@example.com and card is
1234-5678-9876-5432."
}
```

Output:

```
{
    "input_email_body": "Hey, I'm Raj. My email is raj@example.com and card is
1234-5678-9876-5432.",
  "list_of_masked_entities": [
    {
      "position": [26, 43],
      "classification": "email",
      "entity": "raj@example.com"
    },
    {
      "position": [56, 75],
      "classification": "credit_debit_no",
      "entity": "1234-5678-9876-5432"
    }
  ],
  "masked_email": "Hey, I'm Raj. My email is [email] and card is [credit_debit_no].",
  "category_of_the_email": "Request"
}
```

7. Conclusion

The project successfully fulfills the requirements of classifying support emails while protecting sensitive data. The combination of regex and NER provides solid PII coverage, and the API architecture makes the solution production-ready. The app runs locally and on Hugging Face with minimal setup, and the outputs conform exactly to the specified JSON schema.

Submitted by: [Your Name]

Date: April 2025