EMAIL CLASSIFIER AND PII MASKING

22 April 2025

Sejal Vhankade

INTRODUCTION

Emails often carry sensitive Personally Identifiable Information (PII) such as names, phone numbers, credit card details, and government identifiers. In enterprise or customer support settings, classifying emails and ensuring privacy compliance is essential.
This project focuses on building a production-ready application that:

- Masks PII/PCI data using regex and Named Entity Recognition (NER)
- Classifies emails into predefined categories
- Provides the functionality through a clean API
- Is containerized using Docker and deployed via Hugging Face Spaces

APPROACH

To ensure GDPR and data compliance:

- **Regex patterns** are used to detect structured data:
  - Names, email addresses, phone numbers
  - Aadhaar, credit card numbers, CVV, expiry dates, etc.
- **Spacy NER** is used to detect entities like PERSON, ORG, DATE
- Each detected field is masked using ***[FULL_NAME] *** format
- Example:

```
Input: My name is Sejal.

Output: My name is MASKED_NAME
```

B. Email Classification

- A Scikit-learn pipeline was created:
- Preprocessing: Stopword removal, TF-IDF vectorization
- Model: Multinomial Naive Bayes (lightweight and interpretable)
- Classes: ['Work', 'Personal', 'Finance', 'Health', 'Promotions']

IMPLEMENTATION DETAILS

```
.
├── app.py                  # Main Flask app entrypoint
├── api.py                  # API routes and logic
└── data/
   └── emails.csv         # the Data set
├── models.py               # Classification logic and ML model
├── utils.py                # PII masking functions
├── train_model.py          # Script to train and save ML model
├── requirements.txt        # Required Python dependencies
├── README.md               # You're here!
├── space.yaml              # Hugging Face deployment file
├── test_input.json         # Sample input for testing
└── model/
    └── email_classifier.pkl      # Pretrained ML model
```

Project Structure: This is the project structure .

Tech Stack:

- Python 3.10
- Scikit-learn, Spacy, Pandas
- Docker for containerization
- Hugging Face Spaces (Docker SDK) for deployment

MODEL TRAINING AND EVALUATION

- Used a labeled email dataset with categories
- TF-IDF Vectorization with max_features=5000
- Multinomial Naive Bayes achieved ~88% accuracy on test set
- Saved model as model.pkl using joblib

DEPLOYMENT

Hugging Face Deployment

- Steps followed:
- Created a private Hugging Face Space
- Chose Docker SDK and pushed all code with a Dockerfile

- Used:

```
CMD ["python", "app.py"]
EXPOSE 7860
```

- Live inference and testing via /classify API
- Docker Build Command:

```
docker build -t email-classification-api .
```

CHALLENGES AND SOLUTIONS

Challenge	Solution
➔ Regex misses some complex entities	Combined regex + NER using Spacy
➔ Model misclassifies short or ambiguous emails	Added preprocessing and manual tuning
➔ Deployment errors on Hugging Face	Resolved via correct README metadata and Docker base image

HIGHLIGHTS AND BEST PRACTICES

- ✅ Clean project structure with modular files
- ✅ Follows **PEP8** coding guidelines
- ✅ Extensive inline comments for clarity
- ✅ Strict JSON API response format
- ✅ Version control using Git + GitHub
- ✅ Easy-to-replicate and portable using Docker

SAMPLE OUTPUT


```
{
  "category_of_the_email": "General",
  "input_email_body": "Contact Alice Wonderland at alice@wonder.land or +91 98765 43210. DOB: 01/01/1980.",
  "list_of_masked_entities": [
    {
      "classification": "full_name",
      "entity": "Contact Alice Wonderland",
      "position": [
        0,
        24
      ]
    },
    {
      "classification": "email",
      "entity": "alice@wonder.land",
      "position": [
        28,
        45
      ]
    },
    {
      "classification": "phone_number",
      "entity": "+91 98765 43210",
      "position": [
        49,
        64
      ]
    },
    {
      "classification": "dob",
      "entity": "01/01/1980",
      "position": [
        71,
        81
      ]
    }
  ],
  "masked_email": "[full_name] at [email] or [phone_number]. DOB: [dob]."
}
```


CONCLUSION

This project demonstrates proficiency in:

- NLP with regex and NER
- Classification with ML pipelines
- Dockerization and CI-ready deployment
- It showcases full-cycle model development from data cleaning to deployment — ideal for production-facing roles in Data Science, MLOps, or Backend Engineering.