PII-MASKING AND EMAIL CLASSIFIER
ASSIGNMENT REPORT

Introduction:

In modern customer support systems, incoming emails often contain Personally Identifiable Information (PII) such as names, phone numbers, credit card details, and government IDs. Handling such data securely while automating classification into support categories is critical.

Objective:

The objective of this project is to build a **PII-compliant email classification API** that:

1. Detects and masks all PII from raw email text.
2. Classifies the masked email into predefined support categories.
3. Restores the original PII in the final response.
4. Returns results in a strict JSON format via an accessible POST API.

Approach:

PII Detection & Masking:

- **Regex-based Matching**: Designed patterns for email, phone_number, dob, aadhar_num, credit_debit_no, cvv_no, and expiry_no. These are tuned for international and local formats (e.g. +33-1-76-88-45-67, 1234 5678 9123).
- **Named Entity Recognition (SpaCy)**: Used for detecting full_name using en_core_web_sm for reliable NER.
- **Overlap Protection**: Implemented a custom overlap check to ensure no PII entities overwrite each other during masking.

Masked entities are replaced with [entity_type] tags and stored with position metadata for later demasking.

Classification Flow:

- **Input**: Masked subject/body text.
- **Embedding**: sentence-transformers model MiniLM-L6-v2 converts text to semantic vectors.
- **Classifier**: LinearSVC trained on 24,000 email samples to predict:
  - Incident
  - Request
  - Problem
  - Change

Post-classification, original PII is demasked to generate the final output.

Model Selection and Training:

- **SBERT (Sentence-BERT):**
  - Chosen for its balance of **accuracy** and **performance**
  - Paraphrase-MiniLM-L6-v2 provided fast inference and strong semantic understanding.
- **Linear SVM:**
  - Used LinearSVC instead of full SVM for speed and stability on large datasets.
  - Accuracy: ~75% overall, with strong performance on Request, Incident, and Change.
  - Struggled initially with the Problem class due to data imbalance, addressed below.

Training Process Details:

- Dataset: 24,000+ labeled email subjects.
- Embedding: Batched in size 32 for memory safety.
- Split: 80/20 for train/test.
- Evaluation: Macro F1-score + confusion matrix to validate class separation.

Challenges and Solutions:

1. **Overlapping Regex Matches:**

   Issue: 1990 was detected as both dob, cvv, and phone_number.

   Solution: Created priority-based matching order with overlap suppression logic.

2. **Model Performance on Edge Classes:**

   Issue: Problem class had low F1-score.

   Solution: Improved class balance, merged similar subject lines, and optimized embeddings using MiniLM.

3. **Hugging Face Space SDK Limitations:**

   Issue: FastAPI not shown directly in SDK dropdown.

   Solution: Used Static or Docker SDK to override and deploy FastAPI via app.py.