Assignment Report

1. Problem Statement

The objective of this assignment is to design and implement a complete pipeline to automate support email classification. This includes detecting and masking sensitive personally identifiable information (PII) from emails and using machine learning to categorize each email into predefined support categories. The pipeline must expose this functionality through a well-structured REST API.

2. PII Masking Approach

To protect sensitive information, we implemented a hybrid PII masking system:

Regex-Based Entities:

- **Email:** Matches common email formats using `@` and domain structure
- **Phone Numbers:** Indian 10-digit numbers, optionally prefixed with +91 or 0
- **Date of Birth (DOB):** Recognizes formats like DD/MM/YYYY, 15th Aug 1990, etc.
- **Aadhar Number:** 12-digit sequences with optional spaces/hyphens
- **Credit/Debit Card Numbers:** 13–16 digit numeric patterns
- **CVV and Expiry Dates:** Identified by known patterns and masked

spaCy-Based Entities:

- **Full Names:** Extracted using `en_core_web_sm` NER model

Output: The function `mask_pii()` returns two things:

- Masked email with PII replaced using placeholder tags
- List of all masked entities with classification, position, and original text

This satisfies the requirement to remove PII without relying on LLMs.

3. Model: Email Classification Using BERT

We use the transformer-based model `prajjwal1/bert-tiny`, a distilled and efficient version of BERT ideal for local development and limited-resource environments like a Mac M1.

Highlights:

- **Tokenizer:** Hugging Face `BertTokenizerFast`
- **Model:** `BertForSequenceClassification` with `num_labels=4`
- **Training:** Used `datasets.Dataset` from Hugging Face for tokenized inputs, trained over 3 epochs
- **Classes:** Predefined in dataset (e.g., Request, Incident, etc.)

The choice of `bert-tiny` allows fast inference with acceptable accuracy, and the model achieves expected performance on the 24,000-example dataset.

4. API Design

The entire pipeline is exposed through a REST API built with FastAPI.

Endpoint: `POST /classify`

Input: JSON with one field: `email` **Processing:**

- First masks all PII
- Then classifies the masked email

Output:

```
{
  "input_email_body": "...",
  "list_of_masked_entities": [...],
  "masked_email": "...",
  "category_of_the_email": "..."
}
```

Swagger docs are auto-generated at `/docs`.

5. Technical Challenges & Solutions

- **NumPy Compatibility:** Many packages broke when NumPy auto-updated to 2.0+. We pinned it to `1.26.4` to resolve compatibility issues with spaCy, pandas, and transformers.
- **Memory Constraints on Mac M1:** Full-size BERT was too heavy. Switching to `bert-tiny` resolved performance bottlenecks.
- **Tokenizer Overflow:** Addressed max length issues by truncating inputs to 512 tokens.
- **Offline Model Loading:** Allowed for local inference when internet is not available by storing tokenizer/model files locally.

6. Deployment

The application can be deployed easily to any platform that supports FastAPI, including:

- Hugging Face Spaces (`fastapi` interface)
- Render.com (supports Docker-based or web-service deployments)

7. Conclusion

The final system is a robust, efficient, and modular email classification tool that meets all assignment criteria. It ensures compliance by masking PII, categorizes support emails reliably using a fine-tuned transformer, and exposes this pipeline through a modern, testable API interface.

The code is clean, production-ready, and scalable for future extensions such as UI integration, model retraining, or logging enhancements.