Report: Email Classification System with PII Masking

1. Introduction to the Problem Statement

This project uses PII masking to protect data privacy while implementing an automated email classification system that groups incoming support emails into four ITIL-based categories: incident, request, change, and problem. The original email, the masked email, the identified PII entities, and the predicted category are all returned by the system after processing JSON input with email content and masking sensitive information without the use of LLMs.

Masking particular PII types, such as full name, email address, phone number, date of birth, Aadhar number, credit/debit card number, CVV number, and card expiration details, was one of the main requirements. The system is implemented on Hugging Face Spaces by using the Flask API, offering a workable solution for support operations while keeping to standards of privacy.

2. Approach Taken for PII Masking and Classification

2.1 PII Masking Pipeline

The PII masking employs a rule-based approach using Microsoft Presidio enhanced with custom recognizers. The pipeline includes:

Core Components:

- Text preprocessing (HTML removal, whitespace normalization)
- Presidio Analyzer with built-in recognizers for standard PII
- Custom regex patterns for domain-specific entities (Aadhar, card numbers, phone formats)
- Advanced contextual analysis for disambiguation

Key Innovation - CVV Detection: The most challenging aspect was accurately identifying 3-4 digit CVV codes. The solution implements:

- Keyword scanning for CVV-related terms ("CVV", "CVC", "security code")
- Contextual window analysis (±20 words around potential matches)
- Negative filtering for false positives (excluding numbers near "year", "date",

"phone")
- Confidence scoring with base threshold of 0.7 and contextual boosting

Date Disambiguation: Differentiates between Date of Birth, Card Expiry, and general dates using proximity-based keyword analysis:

- Expiry dates: Validated near "expiry", "exp", "expires"
- Date of Birth: Identified through "DOB", "birth", "born"
- General dates: Assigned lower confidence to prevent over-masking

2.2 Email Classification Pipeline

Classification operates on masked email content using a fine-tuned transformer model:

- **Model**: Microsoft DeBERTa-v3-base (multilingual)
- **Process**: Tokenization → Model inference → Softmax activation → Class prediction
- **Categories**: Incident, Request, Change, Problem

3. Model Selection and Training Details

3.1 Model Selection

Two multilingual transformer models were evaluated:

1. **XLM-RoBERTa**: Achieved ~77% accuracy
2. **mDeBERTa-v3-base**: Achieved ~80% accuracy (selected)

mDeBERTa-v3-base was chosen for its superior performance and robust multilingual capabilities.

3.2 Training Configuration

Environment: Google Colab with free GPU resources **Dataset**: Custom-labeled `emails.csv` with 90%/10% train/test split **Training Parameters**:

- 4 epochs with early stopping
- Batch size: 4 per device, 2 gradient accumulation steps
- Mixed precision (FP16) for efficiency

- AdamW optimizer with learning rate scheduling

Class Imbalance Solution: Implemented weighted loss using custom `WeightedTrainer` class with inverse frequency weighting to boost minority classes.

Results: Final model achieved 80% accuracy across all four categories with robust performance on both original and masked text inputs.

4. Challenges Faced and Solutions Implemented

Challenge 1: Accurate CVV Detection *Solution*: Multi-layered contextual analysis with keyword proximity, negative filtering, and confidence thresholding.

Challenge 2: Date Type Disambiguation *Solution*: Context-aware regex patterns distinguishing DOB, expiry dates, and general dates using surrounding keywords.

Challenge 3: Class Imbalance in Training Data *Solution*: Weighted loss function with inverse frequency weighting implemented via custom trainer class.

Challenge 4: Limited Computational Resources *Solution*: Leveraged Google Colab's GPU resources with optimized training strategies (mixed precision, gradient accumulation).

5. System Architecture and Deployment

Technology Stack:

- Backend: Python 3.10, PyTorch, Hugging Face Transformers
- PII Detection: Microsoft Presidio + Custom Regex
- API Framework: Flask
- Deployment: Hugging Face Spaces

API Endpoint: `POST https://milanchndr-email-support-system.hf.space/classify/classify`

Input: JSON with email body

Output: Original email, masked email, PII entities list, predicted category

Sample Response Structure:

json

```
{
  "input_email_body": "Hello, my name is John Doe...",
  "list_of_masked_entities": [
        {"position": [18, 26], "classification": "full_name",
"entity": "John Doe"}
  ],
  "masked_email": "Hello, my name is [full_name]...",
  "category_of_the_email": "Request"
}
```

Conclusion

This project successfully illustrates a combined strategy for privacy protection and email classification. The system offers thorough PII masking for more than eight sensitive data types and achieves 80% classification accuracy. The non-LLM approach is appropriate for production deployment in compliance-sensitive environments since it guarantees deterministic and explicable privacy protection.