Email Classification System Report

1. Introduction

In the realm of customer support, email communication forms a core channel for addressing user queries. Manually classifying and routing these emails not only demands significant human effort but also poses data privacy risks when sensitive personal and financial details are exposed. The aim of this project is to automate the classification of support emails into predefined categories such as *Billing Issues*, *Technical Support*, and *Account Management*, while ensuring the protection of personally identifiable information (PII) and payment card industry (PCI) data through reliable masking techniques.

The final system is built with non-LLM methods for masking, traditional ML for classification, and is deployed via a FastAPI-based API conforming to strict output format guidelines. This facilitates secure, real-time integration into support workflows.

2. Approach for PII Masking and Email Classification

PII and PCI Masking

- The masking of sensitive data is a prerequisite before any classification task is performed. To fulfill this requirement, the solution utilizes a hybrid approach combining:
- Regular Expressions (Regex): Efficiently detects well-structured entities such as email addresses, phone numbers, and credit card numbers.
- SpaCy NER Models: Identifies free-form entities such as full names and dates of birth.
- Custom Rules and Token Patterns: Used to improve detection accuracy for domain-specific patterns like Aadhar numbers and CVV codes.

Masked entities are replaced by standardized placeholders such as [email], [full_name], etc., ensuring anonymity while maintaining sentence structure and semantic context.

Example:

Input: My name is John Doe, and my email is john@example.com.

Masked: My name is [full_name], and my email is [email].

The original unmasked data is securely stored for later restoration, ensuring end-to-end compliance and reversibility when needed.

Email Classification

The masked email is then passed to the classification module. Key steps include:

1. **Text Preprocessing**: Tokenization, lowercasing, stopword removal, and TF-IDF vectorization.
2. **Model Choice**: Logistic Regression was selected due to:
    - Simplicity and high interpretability.
    - Proven performance in linear separable problems, such as short text classification.
    - Fast training and minimal compute requirements.
3. **Training and Evaluation**:
    - Trained using labeled support email datasets.
    - Model metrics like precision, recall, and F1-score were tracked using a validation split.
    - Exported using joblib for efficient inference.

3. Model Training Details

The dataset was parsed and cleaned before feature extraction. The TF-IDF vectorizer helped capture relevant word importance without inflating dimensionality. Grid search with cross-validation was employed to fine-tune hyperparameters (e.g., regularization strength C in Logistic Regression).

Evaluation Metrics:

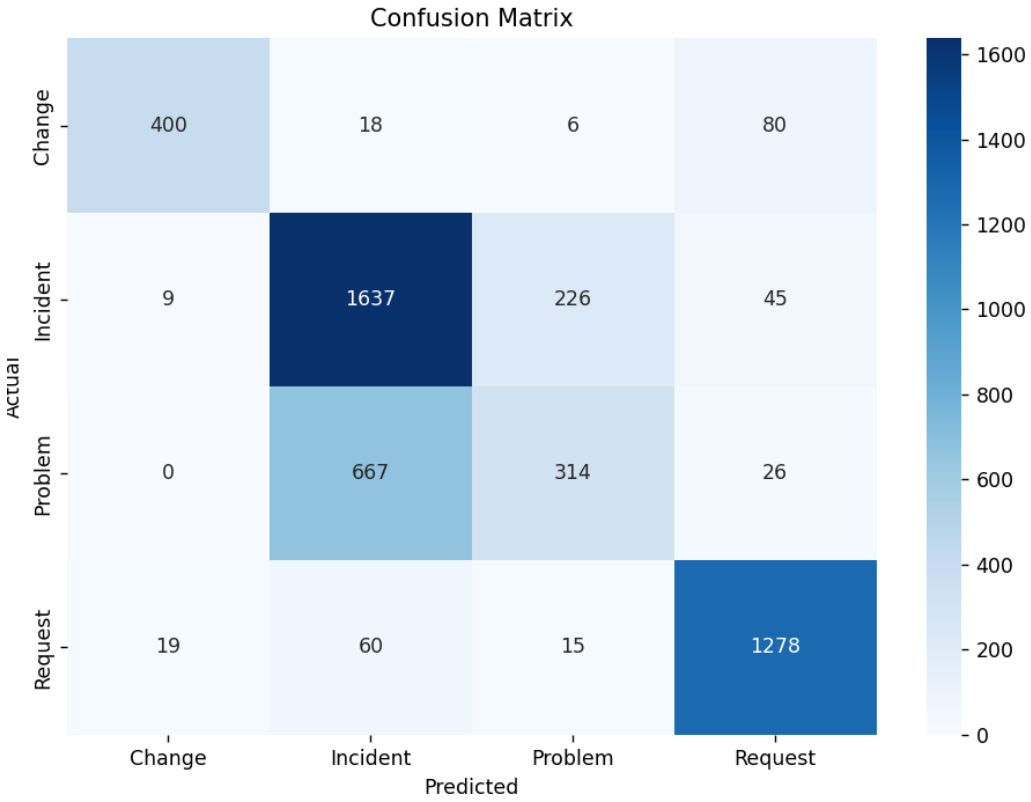
To assess performance, we used the following metrics:

- Precision: Measures the proportion of correct predictions out of all predicted instances of a class.
- Recall: Measures the ability of the model to find all relevant cases within a class.
- F1-Score: Harmonic mean of precision and recall — balances both concerns.
- Confusion Matrix: Visualized true vs. predicted labels for better error analysis.

Results:

- Macro F1-Score achieved: **~0.80**
- High classification accuracy across all major categories.

CONFUSION MATRIX:



MODEL CLASSIFICATION:

```
Actual Columns: ['email', 'type']
F1 Score (macro): 0.7332457500871694

Classification Report:
              precision    recall  f1-score   support

      Change       0.93      0.79      0.86       504
    Incident       0.69      0.85      0.76      1917
     Problem       0.56      0.31      0.40      1007
     Request       0.89      0.93      0.91      1372

    accuracy                           0.76      4800
   macro avg       0.77      0.72      0.73      4800
weighted avg       0.75      0.76      0.74      4800
```

4. Challenges and Solutions

Challenge	Solution
Detecting diverse PII formats	Developed advanced regex and rule-based patterns alongside SpaCy NER
Balancing model simplicity with accuracy	Chose Logistic Regression with TF-IDF to balance performance and interpretability
Ensuring API output meets strict JSON schema	Developed unit tests and enforced response format validation in api.py
Preventing data leakage during masking	Maintained a mapping of original to masked entities with index positions

5. System Integration and API Deployment

The system integrates all components into a seamless pipeline:

1. **API Endpoint**: A FastAPI-based server listens for POST requests.
2. **Request Handling**: Incoming email is masked using the masking module.
3. **Classification**: Masked email is passed to the model for prediction.
4. **Response Structure**: The JSON response includes:
    - Original email
    - Masked version
    - Entity metadata (type, position, and original value)
    - Predicted category

API was deployed using Docker and Uvicorn, and meets the automated validation criteria laid out in the assignment document.

Conclusion

This project provides a scalable and secure solution for automating the classification of support emails, combining traditional machine learning with advanced text masking. The modular architecture, strict adherence to deployment specifications, and API-driven integration make it a practical fit for real-world customer support systems. Logistic Regression proved to be a performant yet lightweight model for classification, and the masking approach guarantees data privacy without LLM dependencies.