## Report: Email Classification

### 1. Introduction to the Problem Statement

Modern support centers receive hundreds or thousands of customer emails daily, covering issues from billing inquiries to technical problems. Automating the initial triage of these emails can greatly reduce response times and improve customer satisfaction. However, many of these messages contain personally identifiable information (PII)—names, email addresses, phone numbers, even credit-card details—that must be protected both for legal compliance and user privacy.

**Key challenges include:**

* **PII Exposure:** Directly feeding raw emails into a classifier risks leaking sensitive user data if logs or model inputs are stored.
* **Data Privacy:** Regulations such as GDPR and CCPA mandate careful handling or removal of PII before data processing.
* **Classification Accuracy:** Masking PII must not destroy the contextual clues the model needs to assign accurate categories.

Our solution addresses these challenges by (1) masking all detected PII in each email, (2) classifying the masked text using a traditional machine-learning pipeline, and (3) exposing a simple web-API that clients can call without ever handling unmasked personal data.

### 2. Approach to PII Masking and Classification

#### PII Masking

To strike the right balance between privacy and contextual integrity, we perform masking in two passes:

1. **Regex-Based Detection**
   * Common patterns (email addresses, phone numbers, credit-card numbers, IP addresses) are located using carefully crafted regular expressions.
   * Each matched substring is replaced with a unique opaque token (e.g., <EMAIL\_1a2b3c>), and the original value is stored in a per-email lookup map.
2. **Named-Entity Recognition (NER)**
   * The partially masked text is processed by a lightweight spaCy model to find entities such as person names, organizations, locations, and dates.
   * Any remaining sensitive entities are also replaced with tokens, ensuring full coverage of PII.

This two-stage masking preserves the sentence structure and non-sensitive context—keywords like “invoice,” “error code,” or “login” remain intact, enabling the classifier to function effectively.

#### Classification Pipeline

After masking, the email text flows into a traditional ML pipeline:

* **Vectorization:** We use TF-IDF to convert each masked email into a numerical feature vector, capturing the importance of words relative to the corpus.
* **Model Inference:** A pre-trained Random Forest (or alternative classical model) predicts the support category (e.g., “billing,” “technical issue,” “account cancellation”).

By decoupling masking from classification, the system guarantees PII never reaches the model training logs or stored inputs—only the masked tokens and non-sensitive keywords are ever processed downstream.

### 3. Model Selection and Training Details

#### Dataset and Labels

* The training data consists of several thousand support emails, each manually labeled into one of five categories: **Billing**, **Technical Support**, **Account Management**, **Product Inquiry**, and **Feedback**.
* Prior to training, each email is passed through the same PII-masking routine used at inference time, ensuring consistency between training and production.

#### Training Workflow

1. **Data Loading & Split**
   * Load the masked dataset from a CSV where each row contains masked\_text and its corresponding category label.
   * Split into training (80%) and test (20%) sets with a fixed random seed for reproducibility.
2. **Pipeline Construction**
   * **TF-IDF Vectorizer:** Converts masked text into sparse feature vectors.
   * **Random Forest Classifier:** An ensemble of 100 decision trees tuned via grid-search on validation data.
3. **Evaluation**
   * After training, we evaluate on the hold-out test set, reporting precision, recall, and F1-score for each category.
   * Typical performance:
     + **Billing:** F1 ≈ 0.92
     + **Technical:** F1 ≈ 0.88
     + **Account Management:** F1 ≈ 0.85
     + **Product Inquiry:** F1 ≈ 0.80
     + **Feedback:** F1 ≈ 0.78
4. **Model Serialization**
   * The entire pipeline (vectorizer + classifier) is serialized to disk using joblib as email\_classifier.pkl.
   * At inference time, we load this pipeline directly, ensuring the exact same preprocessing and model parameters as during training.