1. Introduction to Problem Statement

The objective is to model an email classification system. The system identifies the intent of customer emails while ensuring the personal identity is masked appropriately. The model needs to automatically classify into different identified categories.

2. Approach taken for PII Masking and Classification

PII Masking: The model incorporates a PII masking strategy to protect sensitive information such as name, email, credit card numbers, phone numbers etc. The masking process is carried out in two stages:

1. Regular expression (Regex) to identify commonly used PII types such as email, phone numbers, aadhar, credit card details, phone number etc.
    - Email address that fits usernam@domain.com
    - Phone number that matches 10 digit sequences
    - Aadhar number sequence detection like xxxx xxxx xxxx xxxx
    - Credit card numbers with 16 digits
    - CVV numbers with typical 3 digits
    - Expiry dates in MM/YY or MM-YY
    - Date of birth in formats like DD-MM-YYYY or DD/MM/YYYY or MM-DD-YYY etc.
2. Context-Based Entity Detection with Keywords: To increase accuracy, contextual keyword-based matching was used. This ensures that entities such as phone numbers and email addresses are correctly identified when accompanied by specific keywords (e.g., "phone", "email", "contact").
3. Using re.finditer function, each of these patterns are recognized. When a match is found, it is replaced by a placeholder format [label]. Eg:- [email], [dob] etc. The entity's position and classification is recorded.
4. After applying Regex pattern, SpaCyl's multilingual model is used to detect named entities in the email body. Each identified personal name is replaced by place holder [name].
5. Finally, the function returns masked text that has all PII entities replaced by place holders.

Vectorization of Masked text:

1. TfidVectorizer from scikit-learn is used to convert masked text to numerical vectors.
2. After vectorization, the data is split into features and labels. The features represents the vectorized text i.e., email body, whereas labels represent the type of email.

Model Training:

1. Logistic Regression classifier was selected for email classification due to its simplicity, efficiency, and strong performance with text data. Logistic Regression is particularly suitable for this classification task as it works well with high-dimensional sparse data.

2. The classifier was trained using a labeled dataset containing emails categorized into different types, such as “Incident”, “Request”, etc. The dataset was split into training and test sets in 80:20 ratio.
3. Although Logistic Regression is quite effective without extensive tuning, the maximum iterations parameter was set to 1000 to ensure the model converges effectively.
4. The classifier's performance was evaluated using accuracy, precision, recall, and F1-score. The classification report provided insights into how well the model was performing across different categories of emails.
5. Classification Report:

	precision	recall	f1-score	support
Change	0.95	0.77	0.85	479
Incident	0.69	0.86	0.77	1920
Problem	0.56	0.32	0.41	1009
Request	0.89	0.93	0.91	1392
accuracy			0.76	4800
macro avg	0.77	0.72	0.73	4800
weighted avg	0.75	0.76	0.74	4800

   -

3. Challenges faced and solutions implemented:

1. **PII detection issues:** Detecting PII entities like names or contact details in ambiguous contexts (e.g., initials, partial names) presented challenges in accurate detection. Additionally, initial model was failing due to variation in entities such as same 16-digit code for aadhar and credit card. Aadhar numbers being used with or without spaces or hyphens, credit card numbers without delimiters.
    a. **Solution:** Multiple regular expression patterns were created for each type of PII entity, ensuring that the system could correctly match a wide variety of formats. For example, the Aadhar number regex was updated to capture all variations, whether with or without spaces or hyphens. Similarly, phone numbers were matched with an extended pattern to handle country codes and different separators (e.g., dashes, spaces, parentheses). For partial entity detection, The combined pattern matching approach, which included both the specific PII patterns and keywords that commonly precede these entities, helped improve accuracy.
2. **Language:** Since emails could be written in different languages, initial model failed to detect languages which was present in shared dataset. It also includes different naming convention in date of birth or phone numbers (country code).
    a. **Solution:** To address this, a multilingual SpaCy model (xx_ent_wiki_sm) was employed, capable of recognizing named entities across multiple languages. Additionally, different date formats or numeric separators was also included.
3. **Huggingface Usage:** Since Huggingface deployment was not explored earlier, there were some initial hurdles in understanding deployment. After further exploring different documentations, forums, and videos, the deployment process was studied and solved.