**EMAIL CLASSIFICATION FOR SUPPORT TEAM**

**Introduction:**

Support teams in organizations often receive a large volume of emails that cover a wide range of issues, such as billing queries, technical support, account management, and more. Manually reading and routing these emails to the correct department can be time-consuming and inefficient. To address this, we developed an automated email classification system that categorizes support emails based on their content. Additionally, the system ensures user privacy by detecting and masking Personally Identifiable Information (PII) before processing.

**Objective :**

The goal of this project is to design and implement an email classification pipeline that:

1. Detects and masks sensitive PII such as names, email addresses, phone numbers, and card details.
2. Classifies masked emails into predefined categories.
3. Restores original data after classification.
4. Exposes the entire functionality as a deployable API.

**Approach:**

**PII Masking:**

We used a combination of regular expressions and spaCy's Named Entity Recognition (NER) model to detect the following types of sensitive information:

* Full Name ("full\_name")
* Email Address ("email")
* Phone Number ("phone\_number")
* Date of Birth ("dob")
* Aadhar Card Number ("aadhar\_num")
* Credit/Debit Card Number ("credit\_debit\_no")
* CVV Number ("cvv\_no")
* Card Expiry Number ("expiry\_no")

These entities are replaced with tags (e.g., [email], [phone\_number]) while maintaining their positions, which is necessary for restoring them after classification.

**Email Classification**

For classification, we used a traditional machine learning pipeline:

* **Vectorizer:** TF-IDF
* **Classifier:** Multinomial Naive Bayes

The dataset provided was preprocessed by masking all emails and splitting them into training and testing sets. The classifier was trained on masked data and achieved reliable accuracy across the categories.

**System Integration**

The system consists of four core Python modules:

* utils.py: Contains all utility functions for PII masking
* models.py: Handles model training and loading
* api.py: FastAPI routes and model inference logic
* app.py: Launches the FastAPI server

**API Design**

The API exposes a POST endpoint /classify that accepts the following input format:

{

"email\_body": "string"

}

and returns

{

"input\_email\_body": "original email content",

"list\_of\_masked\_entities": [

{

"position": [start\_index, end\_index],

"classification": "entity\_type",

"entity": "original\_value"

}

],

"masked\_email": "text with PII masked",

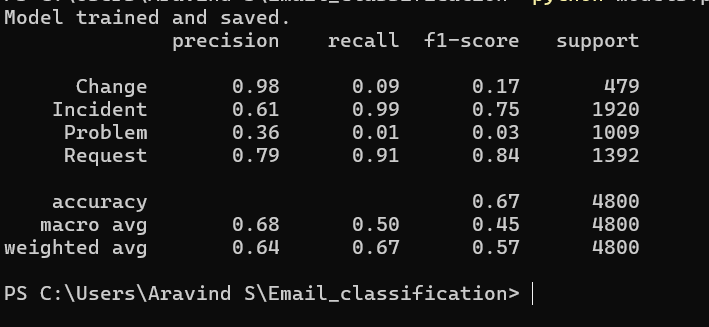
"category\_of\_the\_email": "predicted label"

}

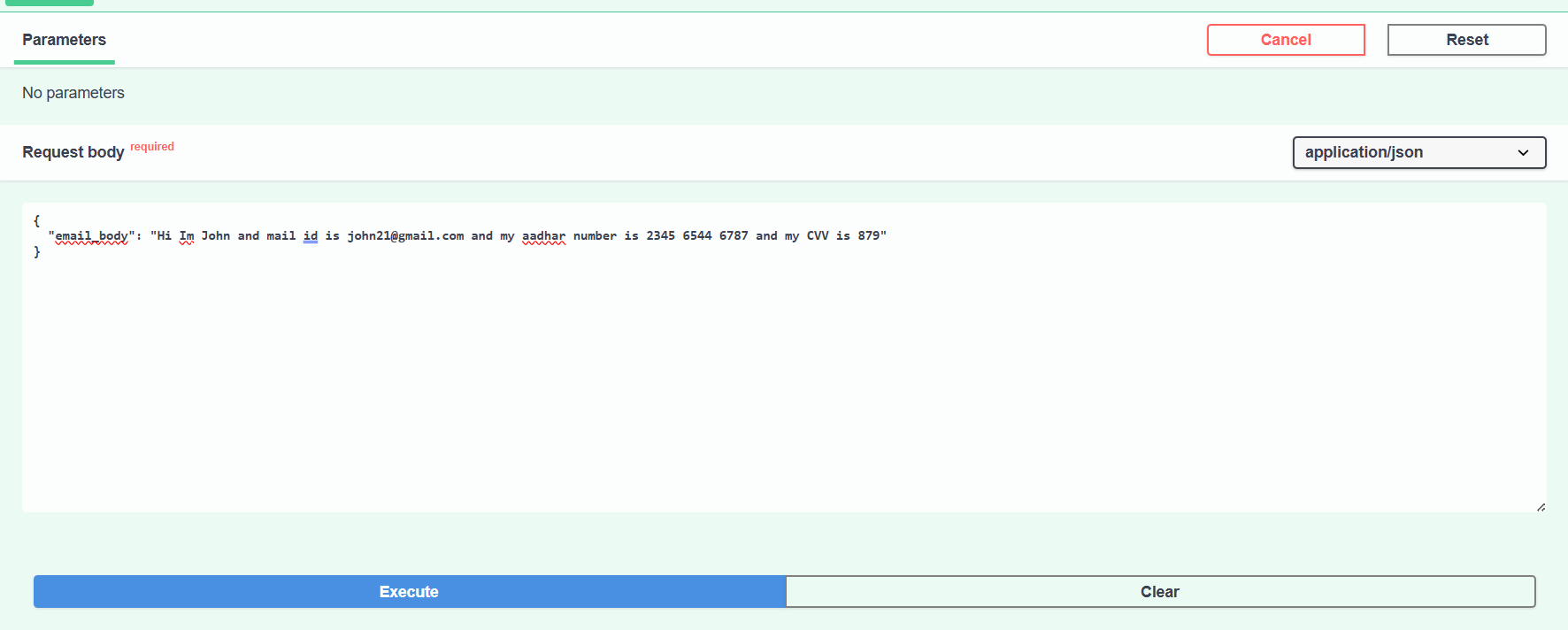
**Result:**

The classifier showed good performance in categorizing emails. The model was trained and evaluated using scikit-learn’s classification report. Here’s where to insert images of results:

**Screenshots:   
  
1. Classification report output:**



**2. Input request in Swagger:**



**3. Classified Output:**