1. Introduction

Customer support teams receive a high volume of emails daily, covering topics such as billing issues, account access problems, and technical support requests. Manually processing and categorizing these emails leads to delays and inefficiencies. Automating this workflow enables faster response times and optimized support staff allocation.

This project addresses the problem by developing an end-to-end system that:

- Automatically detects and masks personally identifiable information (PII) and sensitive card data from support emails.
- Classifies the masked emails into predefined categories using a machine learning model.
- Restores the masked content when required.
- Exposes the entire pipeline through a REST API following a strict output format.

The system ensures privacy compliance and operational efficiency by providing accurate email categorization and data protection.

2. Approach: PII Masking and Classification

The project implements PII masking without relying on Large Language Models (LLMs), using a rule-based approach that combines regular expressions and pattern matching.

PII Entities Detected and Masked:

- `full_name` (e.g., John Doe)
- `email` (e.g., john@example.com)
- `phone_number` (e.g., 9876543210)
- `dob` (e.g., 01/01/1990)
- `aadhar_num` (12-digit Aadhar number)
- `credit_debit_no` (16-digit card numbers)
- `cvv_no` (3-digit number)
- `expiry_no` (MM/YY or MM/YYYY format)

Each occurrence is replaced with a labeled placeholder such as `[email]` or `[credit_debit_no]`, and its original value is stored with start and end positions in the text. This allows easy restoration of data post-classification.

Example:

Original: Hello, I'm John Doe. My email is john@example.com.
Masked: Hello, I'm [full_name]. My email is [email].

After masking, the cleaned email is passed into a text classification model. The final result includes:

- Original input email
- List of masked entities with positions and labels
- Masked email body
- Predicted email category

3. Model Selection and Training Details

This project uses **Logistic Regression** as the classification model to categorize support emails into predefined classes. Logistic Regression was chosen after testing and comparing it with **Multinomial Naive Bayes**, a popular baseline model for text classification.

Why Logistic Regression?

- **Higher accuracy:** Logistic Regression consistently outperformed Multinomial Naive Bayes on the provided dataset.
- **Efficient and easy to deploy:** Simple to implement using `scikit-learn`, and inference is fast and scalable.
- **More robust for real-world use:** Handles sparse and high-dimensional data effectively without overfitting.

Training Process

- **Dataset Used:** `combined_emails_with_natural_pii.csv`
- **Vectorization:** TF-IDF with the following configuration:
  - `stop_words="english"`
  - `ngram_range=(1, 2)`
  - `max_df=0.95`, `min_df=2`
- **Classifier:** `LogisticRegression(max_iter=1000)`
- **Model Saving:** Both model and vectorizer were saved using `pickle` to `classifier_model.pkl`
- **Category Mapping:** Model predictions (e.g., “Incident”, “Problem”) are converted to user-friendly labels like “Technical Support” and “Billing Issues” using a simple mapping dictionary.

This approach resulted in a lightweight, accurate, and deployable solution ideal for integration into a live email classification API.

4. Challenges Faced

1. **Handling Diverse Email Structures**
    - Emails varied significantly in format, structure, and length, making consistent parsing difficult.
2. **Identifying and Masking PII Accurately**
    - Sensitive data like names, phone numbers, emails, and addresses needed to be masked reliably.
    - Regex-based approaches often missed edge cases or over-masked.
3. **Model Overfitting and Generalization**
    - Initial models performed well on training data but poorly on unseen emails.
4. **Deployment Issues with Streamlit**
    - Integrating the classification model with Streamlit UI caused performance bottlenecks.

Conclusion

The project successfully delivered an email classification system with built-in privacy safeguards. The combination of regex-based masking, Logistic Regression-based classification, and structured API response satisfies both accuracy and compliance requirements. It can be deployed in real-world support environments to automate email triaging securely and efficiently.

Check out my project on GitHub:link

Test project on Hugging Face:: link