**Name :P.Nandini**

# mail Classification Project Report

## 1. Introduction

Email classification is crucial for efficiently managing and sorting emails, especially for support teams dealing with large volumes of messages. This project automates the classification of emails into predefined categories such as "Technical Support," "Billing Inquiry," and "General Inquiry," while ensuring that personally identifiable information (PII) is masked to protect user privacy. Additionally, the system is designed to be deployed as a REST API for real-time email classification.

## 2. Approach Taken for PII Masking and Classification

### 2.1 PII Masking

Protecting user privacy is a priority, so PII masking was a key feature of this system. The process involved identifying and masking sensitive data in the emails, such as:

- **Names**
- **Email addresses**
- **Phone numbers**
- **Credit card numbers**
- **Addresses**

PII was masked using regular expressions (regex) and custom detection rules. Any identified sensitive information was replaced with a placeholder like `<MASKED>`, ensuring privacy while preserving the content for classification.

### 2.2 Email Classification

The email classification model was built to automatically categorize emails into predefined classes. The following approach was adopted:

1. **Text Preprocessing**:
    - Tokenization: The email content was split into meaningful tokens.
    - Removal of stopwords: Non-informative words like "the," "is," were removed.
    - Lemmatization: Words were reduced to their base form (e.g., "running" → "run").
2. **Feature Extraction**:
    - **TF-IDF (Term Frequency-Inverse Document Frequency)** was used for feature extraction. This method assigns higher weight to important words, allowing the model to focus on significant terms.
3. **Model Selection**: We used the following components for the classification task:
    - **RF Classifier v3**: A Random Forest classifier that combines multiple decision trees to improve classification accuracy.
    - **LabelEncoder**: This was used to convert categorical email categories into numerical labels for the model.
    - **Vectorizer**: A **TF-IDF Vectorizer** was used to convert email text into a structured format for machine learning.

### 2.3 Model Training

The model was trained on 80% of the data, with 20% used for testing. **RF Classifier v3** was trained on the preprocessed and vectorized email data, with **LabelEncoder** used to transform the email categories into numeric format.

The model achieved an **accuracy of 80%** on the test data, with good performance across additional evaluation metrics such as precision, recall, and F1-score.

## 3. Model Selection and Training Details

### 3.1 Model Selection

The **RF Classifier v3**, an optimized version of the Random Forest algorithm, was chosen for its robust performance in handling large and high-dimensional datasets. This algorithm reduces overfitting by averaging the predictions from multiple decision trees, leading to better generalization.

## 3.2 Training Process

1. **Data Split**:
    - The data was split into a training set (80%) and a testing set (20%).
2. **Training**:
    - The training set was used to train the **RF Classifier v3**, and the email text was vectorized using **TF-IDF** to convert the content into a numerical form.
3. **Model Evaluation**:
    - The model's performance was evaluated using accuracy, precision, recall, and F1-score. It achieved an **accuracy of 80%** on the test set.
4. **Hyperparameter Tuning**:
    - We fine-tuned the hyperparameters using techniques like **GridSearchCV** and **RandomizedSearchCV** to improve performance.

## 3.3 Deployment and API Integration

The trained model was then packaged and deployed as a REST API for real-time email classification using **FastAPI** and **Uvicorn**.

Deployment link :
https://huggingface.co/spaces/nandini2455508/email-classifier-space

- **FastAPI**: We used **FastAPI**, a modern web framework, to create the RESTful API. FastAPI is known for its speed, ease of use, and automatic generation of OpenAPI documentation. It was chosen to ensure high performance and scalability.
- **Uvicorn**: The **Uvicorn** ASGI server was used to serve the FastAPI application. Uvicorn is an asynchronous server that ensures non-blocking I/O operations, making the system efficient and fast for handling real-time email classification requests.
- **Docker**: The application was containerized using **Docker** for consistent deployment across environments.

- **Ngrok**: To expose the local FastAPI server to the internet for testing, **Ngrok** was used. This allowed us to interact with the API in real-time during development.

The REST API accepts email content, performs PII masking, and classifies the email into one of the predefined categories. The response includes the masked email content and the predicted category.

# 4. Conclusion

This project successfully developed and deployed an email classification system that categorizes emails into predefined classes while ensuring that PII is masked to protect user privacy. The system achieves an **accuracy of 80%** and is deployed as a REST API using **FastAPI** and **Uvicorn** for real-time email classification. The key technologies used in the project include:

- **RF Classifier v3** for email classification.
- **LabelEncoder** for encoding categorical labels.
- **TF-IDF Vectorizer** for feature extraction.
- **FastAPI** and **Uvicorn** for API development and deployment.
- **Docker** for containerization and **Ngrok** for testing the API in real-time.

This system can be integrated into real-world applications, enabling automated and secure email classification while ensuring user privacy.