Email Classification API Documentation

1. Overview

This project involves building and deploying a machine learning-based email classifier API capable of identifying the category of an input email while also masking personally identifiable information (PII) from the input. The categories include: Change, Incident, Problem, and Request.

2. Model Performance

The final trained model was evaluated on a test set of 4800 emails. Below are the classification metrics:

```
              precision    recall  f1-score   support

      Change       0.81      0.82      0.81       504
    Incident       0.71      0.79      0.75      1917
     Problem       0.53      0.41      0.46      1007
     Request       0.90      0.91      0.91      1372

    accuracy                           0.75      4800
   macro avg       0.74      0.73      0.73      4800
weighted avg       0.75      0.75      0.75      4800
```

3. API Implementation and Deployment

Framework Used: FastAPI was selected due to its speed, asynchronous capabilities, Pydantic data validation, and built-in Swagger UI.

Endpoint: A single POST endpoint /classify/ was created.

Request:
Accepts a JSON body:

```
{
  "email_body": "string"
}
```

Processing:
1. Receives the input request.
2. Masks any PII via utils.mask_pii(), returning masked_text and list_of_masked_entities.

3. Loads a trained Scikit-learn pipeline from classification_pipeline.joblib via models.load_classifier().
4. Predicts category using models.predict_category() on the masked text.
5. Constructs and returns a structured JSON response.

Response Format:

```
{
  "input_email_body": "string",
  "list_of_masked_entities": [
    {
      "position": [int, int],
      "classification": "string",
      "entity": "string"
    }
  ],
  "masked_email": "string",
  "category_of_the_email": "string"
}
```

Error Handling:
- Returns HTTP 400 for empty inputs.
- Returns HTTP 500 for internal processing errors.

Deployment:
- Containerization: A Dockerfile was used to containerize the application, specifying:
  - Python version
  - Required packages
  - FastAPI and Uvicorn for serving the application
  - Hugging Face Spaces-compatible command:
    CMD ["uvicorn", "api:app", "--host", "0.0.0.0", "--port", "7860"]
- Repository: https://github.com/aaron1-z/email-classification-api
- Deployment Target: Hugging Face Spaces https://huggingface.co/spaces/aaron1z/email-classifier-api

Note: Deployment may not have synced due to Docker SDK issues. Local version is functional.

4. Challenges Faced & Solutions

- Virtual Environment Setup:
  - Initial path resolution issues required using absolute paths.
- CSV Column Mismatch:
  - Modified scripts to handle 'email' and 'type' instead of 'text' and 'category'.
- PII Masking Limitations:
  - Regex masking worked for common PII, struggled with names. Future improvements

suggested.
- Deployment Issues:
  - Hugging Face did not reflect updates due to Docker integration. Manual rebuilds attempted.

5. Final Output

- GitHub Repository: https://github.com/aaron1-z/email-classification-api

- API URL (Hugging Face): https://huggingface.co/spaces/aaron1z/email-classifier-api

Note: Status may vary due to deployment caching issues. Check README for updates.

Endpoint Summary:
- Endpoint: POST /classify/
- Request Body: { "email_body": "..." }
- Response: JSON with input, masked text, entities, and category.

Tested: Fully functional and validated locally with a variety of test cases for classification and PII masking.