Akaike Email Classifier — Final Report

Github Repo: https://github.com/sumitsingh3072/Akaike-Email-Classifier

Introduction to the Problem Statement:

Modern enterprises receive thousands of customers and employee support emails daily. These emails often contain unstructured text and sensitive information, and manually triaging them into categories such as "Incident," "Request," "Problem," or "Change" is both inefficient and error-prone.
The goal of this project is to build an automated, production-ready backend service that:

- Masks personally identifiable information (PII) to ensure privacy,
- Accurately classifies email content into ITSM-relevant support categories, and
- Is easily accessible via a REST API for integration and real-world testing.

Approach Taken for PII Masking and Classification

PII Masking:

- We used spaCy's en_core_web_sm NLP model to detect named entities such as PERSON, ORG, DATE, and GPE.
- Each detected entity is replaced with a generic placeholder (e.g., [name], [organization]).
- The original entities are stored in a list to allow demasking if necessary.

Classification:

- After masking, the cleaned email text is vectorized using TF-IDF to capture text importance.
- A Logistic Regression classifier is then used to predict one of the four classes.
- This classifier was chosen for its speed, interpretability, and strong performance on sparse text data

Model Selection and Training Details:

Dataset:

- A labelled dataset containing email bodies and their corresponding support categories was used.
- It included 4 main classes: Incident, Request, Problem, and Change.

Preprocessing:

- Lowercasing, stopword removal, and lemmatization (via spaCy).
- TF-IDF vectorization with uni- and bi-grams.

Model:

- sklearn.linear_model.LogisticRegression
- Trained with default parameters using 80/20 train/test split

Performance:

- Accuracy: ~76%

- F1 Scores: Incident (0.77), Request (0.91), Problem (0.40), Change (0.85)
- Model saved as email_classifier.pkl and used in the API for real-time inference.

Challenges Faced and Solutions Implemented:

- **PII Masking Errors**: spaCy occasionally missed or misclassified entities. We mitigated this by limiting masking only to key types and using consistent replacement formats.
- **Incorrect Predictions from LLM (Gemini)**: Initially considered using Gemini Pro API for classification, but due to inconsistency and API latency, we reverted to a custom trained model for deterministic performance.
- **FastAPI Deployment**: Deploying on Hugging Face Spaces using Docker required a well-structured repo with app.py, Dockerfile, and a working port (7860). Missing or misnamed files (e.g., Dockerfile) caused build issues which were fixed by careful versioning and dependency resolution.
- **404 Errors on Space URL**: Added a GET / route to serve a welcome message and avoid confusion during browser visits, without affecting the automated grading pipeline which hits only the POST /classify/ endpoint.

API Endpoint Details for Testing:

Main Endpoint: POST https://sumitsingh3072-akaike-email-classifier.hf.space/classify/

Headers: Content-Type: application/json

Sample Request Body:

```
{
  "email_body": "Hi, I cannot access the internal finance dashboard. Please assist."
}
```

Sample Response:

```
{
  "input_email_body": "Hi, I cannot access the internal finance dashboard. Please assist.",
  "list_of_masked_entities": [
    { “position”: [31, 38], “classification”: “resource”, “entity”: "finance” }
  ],
  "masked_email": "Hi, I cannot access the internal [resource] dashboard. Please assist.",
  "category_of_the_email": "Request"
}
```

This endpoint is live, publicly accessible, and supports automated POST-based validation without any frontend dependency.

End of Report