Email Classification System Project Report

1. Introduction

The objective of this project is to build a machine learning pipeline to **classify email data into specific types** — namely **Incident**, **Request**, **Problem**, and **Change** — while ensuring **sensitive information (PII)** such as personal contact details, Aadhar numbers, and names are masked effectively.

The report covers the steps taken to prepare and sanitize the data, apply classification using an SVM model, and evaluate its performance on multilingual email datasets.

---

2. Problem Statement

Sensitive email content often includes PII (Personally Identifiable Information) such as names, email IDs, phone numbers, or national ID numbers. While these entities need to be **protected before training or inference**, they can also interfere with feature extraction.

The challenge was two-fold:

1. **PII masking** using regex with high accuracy, avoiding entity overlap.
2. **Email classification** with noisy, multilingual content using a traditional ML classifier (SVM).

---

3. PII Masking Approach

We avoided using pretrained NER models or LLMs and instead used a **pure regex-based masking system**.

3.1 Regex Strategy

- Masked these patterns:
    - Email addresses
    - Phone numbers (international and Indian formats)
    - Aadhar card numbers (12-digit numeric)
    - Names (when prefixed like "My name is", "I am", etc.)
    - Phrases like "mail me at", "contact:", etc.

3.2 Regex Challenges

- **Overlapping entities**: e.g., My name is John, mail me at john.doe@example.com would cause nested masking.
- **Multilingual tokens** like French (Mon nom est...), German (Mein Name ist...), and Dutch (Mijn naam is...) added complexity.
- Regex was tuned manually, by **trial-and-error**, checking each match against actual sentences from training data.
- Final version used **non-greedy matching**, anchor words, and word boundaries to minimize false positives.

---

4. Classification Model Details

4.1 Model Selection

We chose **SVM (Support Vector Machine)** for its effectiveness in text classification, even with high-dimensional feature space (TF-IDF vectors). It also performs well in class-imbalanced scenarios with correct class weighting.

4.2 Data Distribution

Class	Proportion
Incident	39.94%

Class	Proportion
Request	28.58%
Problem	20.98%
Change	10.48%

This imbalance was addressed using **class weights** and **data augmentation for the minority class** where feasible.

---

5. Model Performance (SVM)

Class	Precision	Recall	F1-Score	Support
Change	0.97	0.81	0.88	504
Incident	0.71	0.91	0.80	1917
Problem	0.67	0.34	0.45	1007
Request	0.92	0.95	0.94	1372

- **Accuracy:** 0.79
- **Macro Avg F1-Score:** 0.77
- **Weighted Avg F1-Score:** 0.77

Observations

- Model is very confident in classifying *Request* emails.
- *Problem* class has low recall (0.34) due to semantic overlap with *Incident*.
- Fine-tuning was done over 3–4 iterations to adapt to multilingual and noisy data.

---

6. Challenges & Fixes

Challenge	Resolution
Overlapping Regex Matches	Refined regex expressions using negative lookahead, grouping, and boundary markers.
Entity Redundancy	Normalized text before masking to reduce pattern mismatch.
Multilingual Inputs (French, German, Dutch)	Extended regex patterns and included international prefixes, phrase detection, and Unicode character handling.
Low performance on "Problem" class	Added manual augmentation of underrepresented class examples, cleaned mislabeled samples.
Space-based NER conflict	Avoided NER altogether; regex with controlled patterns provided better control.

7. API Testing Summary

We created a minimal Flask-based API for testing email classification and PII masking. The service is available on a local endpoint for internal testing.

Endpoint	Method	Description
mask	POST	Accepts raw email text and returns the masked version.
predict	POST	Accepts masked email text and returns predicted class (Incident, Request, etc.).

8. Future Work

- Integrate a **hybrid masking approach** using both regex and light NER where needed.

- Introduce **active learning** to improve recall on low-performing classes like *Problem*.
- Optimize regex patterns using a rules engine.
- Scale the API for production use with containerization.

9. Conclusion

This project demonstrated how to:

- Mask sensitive entities using regex without dependency on large models.
- Train a reliable email classifier on multilingual data using SVM.
- Achieve **79% accuracy**, with strong performance in major classes.

All objectives were met with manual code design, rule-based engineering, and iterative tuning without use of pretrained LLMs.