**Email Classification Support System**

## Problem Statement

The task is to create an email classification system for support team that can perform

* Automatic PII/PCI Data Masking for email entries like name, credit card, CVV etc.
* Classify the masked email into 4 categories – Incident, Problem, Request, Change.
* Maintain proper JSON format architecture to obtain the classified email with properly implemented masked entities.

The project has been created with proper PEP8 guidelines.

There is not cost-embedded model used in the entire project.

## Approach

### Language Detection

The provided dataset is collection of emails with the support types in several languages. Firstly, a language detection logic is implemented for text and identify the different types of languages present in dataset i.e. English, German, French, Portuguese, Spanish, Dutch and Italian.

### Preprocessing and Overview the Dataset

The class label values present in dataset are imbalanced i.e. Incident (maximum having 9586 entries) and Change (minimum having 2516 entries). The languages are also unevenly distributed i.e. English having the most email entries and Italian with the least email entries. On visualizing the dataset, there are no empty data entries and no duplicity. The fields are also properly labelled and have only string values.

### Automatic PII/PCI Masking

All Personally Identifiable Information (PII) and Payment Card Industry (PCI) need to be masked. The particular sections are:

**Personal Identifiers:**

* Full names [full\_name]
* Email addresses [email]
* Phone numbers [phone\_number]
* Dates of birth [dob]
* Aadhar numbers [aadhar\_num]

**Payment Information:**

* Credit/Debit card numbers [credit\_debit\_no]
* CVV codes [cvv\_no]
* Expiry dates [expiry\_no]

To implement this, the technology has been used -

* spacy NER (xx\_ent\_wiki\_sm)
* Refined REGEX patter matching.

So, to obtain proper masking of emails, I implemented spacy NER+Regex approach.

### Language Translation

To handle multilingual language, all the other languages are translated into English language for proper model training. For this, deep\_translator and google\_trans libraries have been used for proper data conversion.

### Model Selection and Training

After careful observation with several traditional ML models, Logistic Regression Model is selected. To resole the issues of class imbalances, chosen fields:

* Class Weights = balanced
* SMOTE – To oversample the minority classes
* TFIDF Vectorization is performed on max\_features = 100000
* Hyperparameter tuning is implemented using GridSearchCV taking F1 micro score for scoring metrics.
* Model training is implemented using scikit-learn train-test-split with proper validation test.

Proper classification report shows good performance:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score |
| Incident | 0.92 | 0.89 | 0.90 |
| Request | 0.85 | 0.88 | 0.86 |
| Problem | 0.83 | 0.79 | 0.81 |
| Change | 0.78 | 0.75 | 0.76 |

The training Macro F1-Score achieved to 0.8456

### Architecture Flow

A screenshot of a diagram

AI-generated content may be incorrect.

### Challenges and Solutions

1. **Multilingual Language Text**:

Used langdetect + spaCy's multilingual NER (xx\_ent\_wiki\_sm) for language-related PII detection

1. **Imbalanced data in context of languages:**

Implemented stratified sampling during train-test split to preserve language distribution ratios

1. **Translation of different languages into English:**

Integrated deep-translator and google-translator with chunked text processing for non-EN → EN conversion

1. **Proper REGEX fine tuning for PII/PCI Masking:**

Developed context-aware and refined regex patterns with iterative testing on multiple test cases

1. **Imbalanced data in context of output labels**

Combined SMOTE oversampling + class-weighted Logistic Regression (class\_weight='balanced')

1. **Model Selection**

Chose Logistic Regression (Open-source) via Scikit-learn, enables <10ms real-time inference, resists overfitting with L2 + GridSearchCV, and handles class imbalance using class\_weight='balanced' and SMOTE.

1. **PEP8 guidelines check**

Automated checks via flake8 + loggings for real-time code quality validation

### Technologies Used

* **Python:** Python 3.10, Pandas, NumPy, Scikit-learn, Regex, NLTK, Spacy
* **Multilingual NLP:** spaCy (xx\_ent\_wiki\_sm), langdetect
* **PII/PCI Masking:** Custom Regex patterns, spaCy NER
* **Language Translation:** deep-translator, ftfy, textwrap
* **Machine Learning:** Scikit-learn, imbalanced-learn (SMOTE), TF-IDF, Logistic Regression, joblib
* **API Deployment**: Flask, Hugging Face Spaces, Docker, Git LFS
* **Code Quality Test:** flake8, PEP8 guidelines