Detailed Report on Akaike Assignment

(Email Classification)

1. **Introduction**

**Problem Statement**

In today's digital landscape, organizations receive thousands of support emails daily, creating significant challenges in email management and privacy protection. The primary problems addressed by this system include:

* **Manual Classification Overhead:** Customer support teams spend considerable time manually categorizing emails into appropriate categories (Incident, Request, Change, Problem), leading to delayed response times and inefficient resource allocation.
* **Privacy and Compliance Risks:** Support emails often contain sensitive personally identifiable information (PII) such as names, email addresses, phone numbers, financial details, and government identification numbers. Exposure of this information poses significant privacy risks and regulatory compliance violations.
* **Scalability Issues:** As organizations grow, the volume of support emails increases exponentially, making manual processing unsustainable and error-prone.

The solution presented is an automated email classification and PII masking system that simultaneously categorizes support emails and protects sensitive information, enabling efficient and secure email processing.

**2. System Architecture and Approach**

**Overall Architecture**

The system is built as a RESTful Flask API with two core components:

* **Email Classification Module:** Utilizes machine learning to categorize emails
* **PII Masking Module:** Employs hybrid detection techniques to identify and mask sensitive information

**PII Masking Approach**

The PII masking system implements a multi-layered detection strategy combining rule-based and machine learning approaches:

* **Regex-Based Detection:**

Implements comprehensive regular expression patterns for structured PII data including email addresses, phone numbers, credit card numbers, CVV codes, expiry dates, Aadhar numbers and date of birth. Provides high precision for well-formatted data with minimal false positives.

* **Named Entity Recognition (NER):**

Utilizes spaCy's pre-trained English model (en\_core\_web\_sm) for intelligent name detection. Employs natural language processing to identify person names in context. Also, Includes fallback regex patterns for name detection when spaCy is unavailable.

* **Hybrid Name Detection:**

Combines spaCy NER results with regex-based patterns to maximize name detection coverage. Implements deduplication logic to prevent overlapping entity detection and Filters out common non-name phrases to reduce false positives.

* **Entity Masking Strategy:**

Replaces detected PII with descriptive placeholders (e.g., [full\_name], [email], [phone\_number]). Maintains text structure and readability while protecting sensitive information and Tracks entity positions for audit and reversal purposes.

**Email Classification Approach**

The classification system employs a supervised machine learning pipeline:

**Text Preprocessing Pipeline:**

* Converts text to lowercase for normalization
* Removes URLs, email addresses, and special characters to focus on content
* Implements tokenization and stopword removal using NLTK
* Applies lemmatization to reduce words to their root forms
* Filters tokens by length to eliminate noise

**Feature Engineering:**

* Utilizes TF-IDF (Term Frequency-Inverse Document Frequency) vectorization
* Implements n-gram analysis (unigrams and bigrams) for context capture
* Applies feature selection with max\_features=10000 and document frequency filtering
* Uses scikit-learn's built-in English stopwords for additional noise reduction

**Classification Algorithm:**

* Employs Random Forest classifier for robust multi-class classification
* Implements class balancing to handle imbalanced datasets
* Uses ensemble learning with 100 estimators for improved accuracy and generalization

1. **Model Selection and Training Details**

**Algorithm Selection**

**Random Forest Selection:**

* **Robustness:** Handles noisy text data and missing features effectively.
* **Interpretability:** Provides feature importance insights for model understanding.
* **Performance:** Demonstrates excellent performance on text classification tasks.
* **Scalability:** Supports parallel processing and handles large feature spaces efficiently and prevents over-fitting.

**Training Process**

**Data Preparation:**

* Implements stratified train-test split (80-20) to maintain class distribution
* Validates dataset integrity with missing data removal
* Performs class distribution analysis to identify potential imbalances

**Model Configuration:**

Pipeline([

('tfidf', TfidfVectorizer(

max\_features=10000,

ngram\_range=(1, 2),

min\_df=2,

max\_df=0.95,

stop\_words='english'

)),

('classifier', RandomForestClassifier(

n\_estimators=100,

random\_state=42,

class\_weight='balanced',

n\_jobs=-1

))

])

**Evaluation Metrics:**

* Accuracy scoring for overall performance assessment
* Detailed classification report with precision, recall, and F1-scores
* Confusion matrix analysis for class-specific performance evaluation

**Model Persistence and Loading**

* Implements joblib serialization for efficient model storage
* Supports multiple model path locations for deployment flexibility
* Includes model validation and loading error handling
* Provides graceful fallback to rule-based classification when models are unavailable

1. **Challenges Faced and Solutions Implemented**

**PII Detection Challenges**

**Challenge:** Handling diverse PII formats and contexts

* **Solution:** Implemented comprehensive regex patterns with flexible matching
* **Example:** Phone number pattern (?:\+91[-.\s]?)?(?:\d{10}|\d{3}[-.\s]?\d{3}[-.\s]?\d{4}) handles multiple Indian and international formats

**Challenge:** Name detection accuracy and false positives

* **Solution:** Developed hybrid approach combining spaCy NER with regex patterns
* **Implementation:** Created exclusion lists for common phrases and implemented position-based deduplication

**Challenge:** spaCy model dependency and deployment issues

* **Solution:** Implemented graceful fallback mechanisms with try-catch blocks
* **Benefit:** System remains functional even without advanced NLP dependencies

**Classification Challenges**

**Challenge:** Model availability and deployment reliability

* **Solution:** Developed robust fallback classification system using keyword-based rules
* **Implementation:** Created rule-based classifier that maps keywords to categories with confidence scores

**Challenge:** Text preprocessing consistency and performance

* **Solution:** Implemented comprehensive preprocessing pipeline with error handling
* **Features:** Handles empty text, encoding issues, and NLTK dependency failures

**Challenge:** Model loading and path management

* **Solution:** Implemented multi-path model loading with priority-based fallback
* **Paths:** Checks multiple common locations (models/, ./, current directory)

**System Reliability and Error Handling**

**Challenge:** API reliability under various failure conditions

* **Solution:** Comprehensive error handling throughout the system
* **Implementation:** 
  + Try-catch blocks around all major operations
  + Detailed logging for debugging and monitoring
  + Graceful degradation with fallback mechanisms
  + Proper HTTP status codes and error messages

**Challenge:** Performance optimization for real-time processing

* **Solution:** Optimized preprocessing and prediction pipeline
* **Features:** 
  + Efficient regex compilation and reuse
  + Streamlined text processing pipeline
  + Minimal memory footprint for API deployment

**Deployment and Production Considerations**

**Challenge:** Dependency management across different environments

* **Solution:** Comprehensive requirements specification and environment handling
* **Implementation:** Clear dependency versioning and optional component loading

**Challenge:** API documentation and usability

* **Solution:** Implemented comprehensive endpoint documentation
* **Features:** Health check endpoints, usage examples, and clear error messages

5. Conclusion

The implemented email classification and PII masking system successfully addresses the core challenges of automated email processing while maintaining high accuracy and reliability. The hybrid approach to PII detection achieves comprehensive coverage, while the machine learning-based classification provides accurate categorization with robust fallback mechanisms.

**Key achievements include:**

* **Comprehensive PII Protection:** Detects and masks 8 types of sensitive information
* **Accurate Classification:** Provides reliable email categorization with confidence scoring
* **Production Ready:** Implements robust error handling and fallback mechanisms
* **Scalable Architecture:** Designed for high-volume email processing
* **Privacy Compliant:** Ensures sensitive information protection throughout the process