**Employee Data Field Classifier – Workflow Documentation**

## 1. Introduction

The ML Field Classifier automatically identifies emails, phone numbers, and addresses in employee datasets. It reduces manual effort and improves data consistency across multiple sources.

## 2. Sample Dataset

| Contact Info | Phone | Address |
| --- | --- | --- |
| john.doe@company.com | (555) 123-4567 | 123 Elm Street, NY |
| (555) 123-4567 | N/A | 456 Oak Ave, CA |
| 123 Elm Street, NY | 555-987-6543 | 789 Pine Rd, TX |
| jane\_smith@org.net | 123-456-7890 | 321 Maple Blvd, WA |
| 987-654-3210 | N/A | N/A |

## 3. Method 1: Regex + ML Classifier (Random Forest)

**Workflow Steps:**

1. Data Input: Upload CSV or Excel file.

2. Feature Extraction: - Presence of @ → potential email - Digits → possible phone or address - Phone pattern → phone - Text length → helps identify addresses

3. Prepare Training Data: Label each cell.

4. Model Training: Train Random Forest classifier.

5. Prediction: Apply model to new/unlabeled cells.

6. Output: Map predictions back to columns.

**Pros:** Fast, interpretable

**Cons:** Requires labeled data, cannot handle fuzzy/unstructured fields

## 4. Method 2: Transformer-Based Classification (Sentence-BERT + KMeans)

**Workflow Steps:**

1. Data Input: Flatten dataset.

2. Embedding: Convert text to semantic vectors using Sentence-BERT.

3. Clustering: Apply KMeans to group similar vectors.

4. Cluster Labeling: Assign labels to clusters (email, phone, address).

5. Prediction: Embed new data and assign to nearest cluster.

**Pros:** Handles fuzzy/unstructured fields

**Cons:** Small datasets unstable, requires manual cluster labeling

## 5. Method 3: Embedding + Vector Search (Qdrant-style)

**Workflow Steps:**

1. Data Input: Flatten dataset.

2. Vectorization: Create embeddings with Sentence-BERT.

3. Store in Vector DB: Save embeddings.

4. New Data Processing: Embed new text → query DB → find closest embedding.

5. Label Assignment: Assign field type based on closest match or heuristic.

**Pros:** Scalable, production-ready, integrates with LangChain/Qdrant

**Cons:** Needs reference prototypes

## 6. Method 4: Rule-Augmented Hybrid

**Workflow Steps:**

1. Data Input: Flatten dataset.

2. Rule Matching: - Contains @ → email - Matches phone regex → phone - Contains keywords (Street, Ave, Blvd) → address

3. Fallback: Anything not matching rules → unknown.

4. Optional ML: Apply ML for ambiguous cases.

**Pros:** Fast, simple, interpretable

**Cons:** Rules can fail on unusual/new formats

## 7. Comparison Table

| Method | Pros | Cons | Speed (sec/sample) |
| --- | --- | --- | --- |
| Regex + Random Forest | Fast, interpretable | Needs labeled data | 0.012 |
| Sentence-BERT + KMeans | Handles fuzzy fields | Small datasets unstable | 0.072 |
| Embedding + Vector Search | Scalable, production-ready | Needs prototypes | 0.031 |
| Rule-Augmented Hybrid | Simple, very fast | Rules can be brittle | 0.001 |

## 8. Recommended Approach for Large Datasets

For **very large datasets**, the best approach is **Embedding + Vector Search (Qdrant-style)** because:

- It scales efficiently with millions of rows.

- Vector databases support **fast nearest-neighbor search**.

- Integrates well with production pipelines (LangChain, Qdrant).

- Can handle messy/unstructured data with semantic embeddings.

Optional: Use **hybrid rules** for extremely common patterns (emails/phones) to speed up processing further.