**RPM Categorization Project Documentation**

### **Project Overview**

The RPM Categorization pipeline is a multi-step solution designed to classify ERP component data into meaningful product categories and subcategories. The classification is performed using a hybrid approach of rule-based direct mapping and machine learning-based inference, ensuring scalability, robustness, and traceability.

This solution is designed to process high-volume manufacturing or procurement data, ensuring consistent categorization across material records, and enabling downstream analytics such as spend analysis, sourcing optimization, and inventory control.

### **Pipeline Stages**

#### **1. Data Extraction and Key Generation**

**Script:** step\_1\_extract\_and\_merge.py

* **Purpose:**
  + Extract required columns from input .parquet files.
  + Identify and cache unique component records using SQLite to ensure deduplication.
  + Generate keys for downstream modeling.
* **Detailed Steps:**
  + Reads .parquet files containing ERP component data.
  + Uses fields like CMPNT\_MATL\_NUM, CMPNT\_MATL\_DESC, CMPNT\_MATL\_TYPE\_CD, CMPNT\_CAT\_CD\_DESC, CMPNT\_UOM\_CD, and LOGL\_KEY\_COMB\_COL\_VAL.
  + Applies AI override if valid predictions exist from prior runs.
  + Direct mappings are performed using provided CSV mapping files on fields like CMPNT\_CAT\_CD\_DESC and CMPNT\_MATL\_TYPE\_CD.
  + New keys are written into key\_output folder for model inference.
* **Output:**
  + key\_output: Parquet files containing unique, previously unseen logical keys.
  + final\_output: Files with direct-mapped or AI-overridden values and flags for ML inference.

#### **2. Preprocessing and Feature Engineering**

**Script:** step\_2\_preprocess.py

* **Purpose:**
  + Clean and standardize text fields.
  + Map unit of measure and material type into engineered categorical features.
* **Detailed Steps:**
  + Applies text cleaning: lowercase, punctuation removal, spacing normalization, duplicate removal.
  + Expands common abbreviations using a user-provided abbreviation map.
  + Maps CMPNT\_UOM\_CD into UNIT\_GROUP (e.g., KG -> CHM, EA -> Discrete).
  + Maps CMPNT\_MATL\_TYPE\_CD to domain-specific categories such as FINISHED\_PRODUCT, RAW\_MATERIAL, etc.
* **Generated Features:**
  + CMPNT\_MATL\_DESC\_CLEAN: cleaned description text.
  + CMPNT\_MATL\_DESC\_LEN: character length of description.
  + UNIT\_GROUP, CMPNT\_MATL\_TYPE\_CATEGORY: categorical mappings for modeling.
* **Output:**
  + .parquet files with additional feature columns ready for inference.

#### **3. Inference (Category and Subcategory Prediction)**

**Script:** step\_3\_inference\_run.py

* **Purpose:**
  + Predict missing product categories and subcategories for unmapped rows.
* **Detailed Steps:**
  + Loads pretrained LightGBM models for category and subcategory prediction.
  + Uses SentenceTransformer (BioBERT) to create embeddings from cleaned descriptions.
  + Combines embeddings with engineered features (length, unit group, material type).
  + Applies separate subcategory models for CHM, PKG, and FNW categories.
  + Fallback strategy for other categories using same label as subcategory.
* **Inference Logic:**
  + AI\_FINAL\_CATEGORY\_CONFIDENCE and AI\_MATCHING\_REASON\_FINAL\_CATEGORY are logged for traceability.
  + Rows with low model confidence (< 0.6) are marked as Other.
* **Output:**
  + Final .parquet files with predicted AI\_FINAL\_CATEGORY and AI\_FINAL\_SUBCATEGORY.

#### **4. Merge Predictions with Key Files**

**Script:** step\_4\_merge\_with\_key\_files.py

* **Purpose:**
  + Consolidate predictions from all files.
  + Merge global predictions with key datasets for traceable output.
* **Detailed Steps:**
  + Reads all .parquet outputs from inference step.
  + Validates presence of required AI prediction columns.
  + Concatenates valid predictions and deduplicates on CMPNT\_MATL\_NUM by choosing the prediction with highest confidence.
  + Saves the consolidated global\_predictions.parquet file.
  + Iteratively joins predictions with original key\_output files.
  + Adds a File\_Name column for traceability.
* **Final Merging Logic:**
  + Ensures null CMPNT\_MATL\_NUM rows don’t retain any AI predictions.
  + Removes intermediate global predictions file after merge to optimize disk usage.
* **Output:**
  + Updated .parquet files in final\_output\_dir/ with AI predictions merged into original keys.

### **Key Innovations**

* **Hybrid Mapping Approach:** Direct rule-based mapping first, followed by ML prediction only where mapping fails or confidence is low.
* **Disk-based Deduplication with SQLite:** Prevents memory overhead during multiprocessing using efficient WAL-mode SQLite caching.
* **Modular Design:** Scripts can be executed independently in any cloud or local environment with parallel processing support.
* **Explainability and Auditability:** Matching reason fields and confidence scores help track each prediction’s source and reliability.

### **Directory Structure**

project\_root/  
├── input\_data/ # Raw input files in .parquet format  
├── mapping\_files/ # Rule-based category mapping files (.csv)  
├── abbreviation\_map/ # CSVs for abbreviation expansion  
├── key\_output/ # Generated keys for inference  
├── final\_output\_preprocessed/ # Direct-mapped files + feature engineered  
├── final\_output\_predicted/ # Final output after ML inference  
├── final\_output\_merged/ # Final merged output with key files

### **Execution Summary**

# Step 1: Run rule-based mapping and generate keys  
python step\_1\_extract\_and\_merge.py \  
 --input\_path input\_data/ \  
 --mapping\_csv mapping\_files/ \  
 --key\_output key\_output/ \  
 --final\_output final\_output\_preprocessed/  
  
# Step 2: Preprocess files with abbreviation expansion and feature creation  
python step\_2\_preprocess.py \  
 --input\_path final\_output\_preprocessed/ \  
 --abbrev\_map abbreviation\_map/ \  
 --output\_path final\_output\_preprocessed/  
  
# Step 3: Perform inference on unmapped rows  
python step\_3\_inference\_run.py \  
 --input\_path final\_output\_preprocessed/ \  
 --final\_output\_dir final\_output\_predicted/  
  
# Step 4: Merge model outputs with original key data  
python step\_4\_merge\_with\_key\_files.py \  
 --inference\_output\_dir final\_output\_predicted/ \  
 --key\_output\_dir key\_output/ \  
 --final\_output\_dir final\_output\_merged/

### **Dependencies**

* Python >= 3.8
* pandas, numpy, lightgbm, sentence-transformers, joblib
* AzureML SDK for model retrieval

### **Future Enhancements**

* Integrate active learning loop for human-in-the-loop review.
* Build confidence threshold calibration dashboards.
* Real-time web service or batch scoring via Azure ML pipelines.
* Enable category-specific rules via config-driven YAML/JSON logic.

### **Contact**

* **Owner:** Arvind
* **Use Case:** ERP component categorization for better procurement and inventory insights.
* **Stack:** Python, Pandas, LightGBM, BioBERT, AzureML, SQLite
* **Deployment Target:** Azure ML Pipelines / Local / Batch Cloud Compute

**End of Document**