

# Amazon ML Challenge

Pritish Saha

October 2025

## 1 Introduction

This document provides a complete, step-by-step plan to build a winning solution for the Smart Product Pricing Challenge. The objective is to predict product prices from multimodal inputs — the concatenated catalog text (title, description, quantity lines) and the product image — under the competition’s constraints: use only the provided training data, obey the SMAPE evaluation metric, and ensure the final model is under 8B parameters and licensed MIT/Apache-2.0. We assume access to an NVIDIA A100 40GB (SXM4), enabling both parameter-efficient and full fine-tuning where appropriate.

## 2 Problem Definition and Constraints

- **Task:** Regress product price from (i) text field `catalog_content` and (ii) `image_link`.
- **Metric:** Symmetric Mean Absolute Percentage Error (SMAPE):

$$\text{SMAPE} = \frac{1}{n} \sum_{i=1}^n \frac{|\hat{y}_i - y_i|}{(|y_i| + |\hat{y}_i|)/2} \times 100 \%$$

- **Compliance:** Strictly no external price lookup (no web scraping/APIs/manual lookup). General-purpose pretrained encoders with permissive licenses are allowed.
- **Model/License:** Final models must be MIT or Apache-2.0 licensed and have fewer than 8B parameters.
- **Data:** `train.csv` (~75k), `test.csv` (~75k); columns: `sample_id`, `catalog_content`, `image_link`, `price` (train only).

## 3 High-Level Approach

We train two complementary models and ensemble them with stacking to minimize SMAPE:

1. **Model A (Feature + GBDT)**: Rich engineered features from text (regex quantity/units, brand, claims) and OpenCLIP image embeddings, trained with LightGBM on log-price with monotone constraints.
2. **Model B (VLM DDP)**: Qwen2.5-VL-3B-Instruct fine-tuned via Unsloth LoRA with DDP/WebDataset streaming (offline JSONL→tensors→shards, chat templating, special-token masking fix), optimized for price extraction from image+text.

We blend OOF predictions using a meta-learner. This pipeline targets strong leaderboard performance with robust generalization and SMAPE-centric optimization.

## 4 Environment, Resources, and Reproducibility

- **Hardware**:  $1 \times$  A100 40GB. Optional CPU workers for I/O.
- **Frameworks**: PyTorch, LightGBM. All MIT/Apache/BSD style.
- **Licenses and Models** (permissible):
  - Qwen/Qwen2.5-VL-3B-Instruct (Apache-2.0): <https://huggingface.co/Qwen/Qwen2.5-VL-3B-Instruct>
  - OpenCLIP ViT-L/14 (Apache-2.0) for image embeddings (Model A features).
  - LightGBM (MIT). Tesseract (Apache-2.0).
- **Determinism**: Fix seeds, record package versions, save checkpoints, persist CV splits, cache embeddings to disk, log configuration files.

## 5 Data Ingestion and Image Acquisition

### 5.1 Text Loading

Load `train.csv` and `test.csv`. Deduplicate via `sample_id`, ensure non-empty `catalog_content`. Create a working copy and a lower-cased copy for parsing.

### 5.2 Image Download (Amazon CDN hardening)

Product image URLs point to the Amazon media CDN (e.g., <https://m.media-amazon.com/images/I/71XfHPR36-L.jpg>). Implement a robust fetcher:

- Use a shared keep-alive HTTP session with realistic headers (User-Agent, Accept, Connection). Limit concurrency to 16–32 to avoid throttling; retry with exponential backoff on 429/5xx.
- Validate response: status 200, `Content-Type: image/*`. Decode with PIL, apply EXIF transpose, convert to RGB.

- Save images deterministically as `images/<sample_id>.jpg` to avoid filename collisions. Maintain a `missing_image` flag when retrieval/decoding fails.
- Preprocess for encoders: for OpenCLIP, resize long side to 336 and center-crop to 336×336; for Qwen2-VL, resize to max side 448 and pad as required by its processor.

### 5.3 Caching and Resilience

Cache images and embeddings on disk (e.g., `parquet/npz`). Use a failure queue for retrying bad URLs. Keep a manifest (CSV) of image status per `sample_id`.

## 6 VLM-Prompted Structured Extraction

### 6.1 Input Preparation

Strip HTML tags/BRs and normalize whitespace on `catalog_content` (light cleaning only). Images are downloaded and resized as per the model processor.

### 6.2 Prompt and Processing

We pass the image and cleaned catalog text to Qwen2.5-VL using a consistent chat template. The prompt requests a concise JSON-like output with fields: `brand`, `pack_count`, `mass_g`, `volume_ml`, `category`, and `keywords` (list). The ‘AutoProcessor’ formats the multimodal input; Unsloth SFT handles training-time collation.

### 6.3 Parsing and Validation

Model outputs are parsed with strict guards: numeric validation for quantities (non-negative, plausible ranges), unit normalization (g/ml), and brand hygiene (trim/punct/whitespace). When an extracted field is missing, we fall back to simple heuristics on catalog text (e.g., picking the first plausible brand token), but we avoid rule-based regex pipelines or OCR; the VLM remains the primary source of structured attributes.

### 6.4 Pseudo-Category via Clustering

Compute sentence embeddings (e.g., DeBERTa or TF-IDF/SVD signals) and cluster with MiniBatchKMeans (e.g., K=300–600). Use cluster id as a pseudo-category for downstream aggregation and meta-learning.

## 7 Representations and Features

### 7.1 Text Representations

- **TF-IDF** for Model A: word uni/bi-grams and char 3–5-grams with capped vocabulary and optional truncated SVD to 2–8k dims.

### 7.2 Image Representations

- **OpenCLIP ViT-L/14 (336px)** pooled embeddings for Model A features (Apache-2.0).
- **Qwen2.5-VL** native visual encoding via its processor for Model B.

### 7.3 OOF Aggregates (Leakage-safe)

Using training folds only, compute per-entity aggregates and bring them back as features via out-of-fold (OOF) predictions to prevent leakage:

- brand median/mean price; pseudo-category median; normalized unit medians (\$/100g, \$/100ml, \$/each).
- Top-token medians for frequent informative tokens (e.g., “organic”, “keto”).

## 8 Models and Training

### 8.1 Model A: LightGBM on Engineered Features

- **Inputs:** structured features (brand/units/packs/claims), TF-IDF/SVD, OpenCLIP embeddings, OOF aggregates, and `missing_image`.
- **Target:**  $\log(\text{price} + 1)$ . Loss: MAE or Huber on log-price; monitor SMAPE on price-space for early stopping.
- **Monotone constraints:** enforce that price increases with `total_mass_g`, `total_volume_ml`, `total_count_each`.
- **Hyperparams:** `num_leaves` 512–2048, `feature_fraction` 0.6–0.9, `learning_rate` 0.02–0.06, early stopping on validation SMAPE.

### 8.2 Model B: Qwen2.5-VL-3B (DDP/WebDataset)

- **Model:** Qwen/Qwen2.5-VL-3B-Instruct (Apache-2.0), fine-tuned via Unsloth LoRA (4-bit base, bf16) with DDP.
- **Offline preprocessing:** Create chat JSONL (system+user with image+text; assistant price for train). Apply ‘AutoProcessor’ with chat template to produce tensors: `input_ids`, `attention_mask`, and patch embeddings `pixel_values` with `image_grid_thw`. Save `.npy` per sample and shard into WebDataset TARs.

- **Streaming:** Use WebDataset to stream shards efficiently (sequential tar I/O). Decode `.npy` blobs, validate patches =  $t \times h \times w$ . Shuffle buffers at shard/sample level.
- **Training:** Unsloth SFT with custom collator: pad, build labels, and **mask all special tokens**  $\geq$  vocab size to  $-100$  to avoid out-of-bounds label indices. Disable gradient checkpointing when VRAM allows; tune batch/num workers.
- **Checkpointing/Debug:** Periodic step-based checkpoints (rank-0 only with DDP barrier). Debug cells to verify shapes and token ranges; subset SMAPE monitoring callback for quick pulse.
- **Inference:** Stream test shards; decode generated tokens; extract numeric price from the generated tail (post prompt), with fallbacks; assemble `test_out.csv`.

## 9 Cross-Validation and Stacking

### 9.1 Cross-Validation

Use 5–10 folds stratified by price quantiles and balanced across pseudo-category and brand-frequency buckets to stabilize distributional coverage.

### 9.2 Stacking

- Produce OOF predictions from Models A and B.
- Train a meta-learner (Ridge/ElasticNet/LightGBM) on OOF predictions plus a few global features (e.g., unit totals) to predict price on validation.
- Blend test predictions fold-wise to obtain the final submission.

## 10 SMAPE-Centric Target Handling and Losses

- Train in log-price space to stabilize heavy-tailed distributions:  $t = \log(\text{price} + 1)$ .
- For neural models, compute a SMAPE-approximate loss on price-space with a small  $\varepsilon$  to stabilize denominators; clamp predictions to non-negative during loss.
- For GBDT, MAE/HUBER on log-price with SMAPE monitored for early stopping and model selection.

## 11 Post-Processing and Safety Checks

- Enforce non-negativity:  $\hat{y} = \max(0.99, \hat{y})$ .
- Cap extreme outliers (e.g., 99.8th percentile); verify shape against validation targets.
- Consistency: if both `total_mass_g` and `total_volume_ml` are zero, but `count/pack` cues exist, rely more on count-based priors and down-weight uncorroborated visual extremes.

## 12 Inference Pipeline

1. Load cached text parses, embeddings (OpenCLIP/DeBERTa), and image status. For missing images, insert learned fallback embeddings and set `missing_image=1`.
2. Compute features for Model A; run Models A and B to obtain predictions.
3. Apply meta-learner stacking to combine base predictions.
4. Clamp, clip extremes, and write exactly `dataset/test_out.csv` with columns `sample_id,price` for all test rows, preserving order or matching on id.

## 13 Ablation and Sanity Checks

- Validate that unit normalization and pack parsing reduce error notably on categories with multi-pack listings.
- Check image-only vs text-only vs multimodal deltas to ensure both modalities add value.
- Verify OOF aggregate features are leakage-safe (computed within folds only).
- Compare QLoRA vs full finetune for Qwen2-VL on A100 40GB; prefer the best SMAPE with stability.

## 14 Timeline (Executable Plan)

1. **Day 1:** Implement hardened image downloader; text cleaners; quantity/unit/pack parsers; unit normalization; compute TF-IDF; run OpenCLIP and DeBERTa embeddings; cache everything; train Model A baseline with CV.
2. **Day 2:** Train Model B (two-tower) with SMAPE-approx loss; produce OOF; compute OOF aggregates; begin QLoRA on Qwen2-VL; evaluate per-fold.
3. **Day 3:** Train stacking meta-learner; finalize inference; generate submission.

## 15 Compliance and Documentation

- No external price lookup/scraping. Only general-purpose pretrained models (MIT/Apache-2.0) are used.
- Document methodology, models, feature engineering, and validation steps in the required 1-page format; include model licenses.
- Cite the Amazon media CDN example image host in the README and ensure downloader robustness: <https://m.media-amazon.com/images/I/71XfHPR36-L.jpg>.

## 16 Expected Outcomes

By explicitly modeling unit/pack quantities, leveraging both text and visual cues via three complementary learners, optimizing training and calibration for SMAPE, and enforcing robust inference safety checks, this pipeline aims to achieve top-tier leaderboard performance while satisfying all challenge constraints.

## 17 Implementation-Level Engineering Diagram

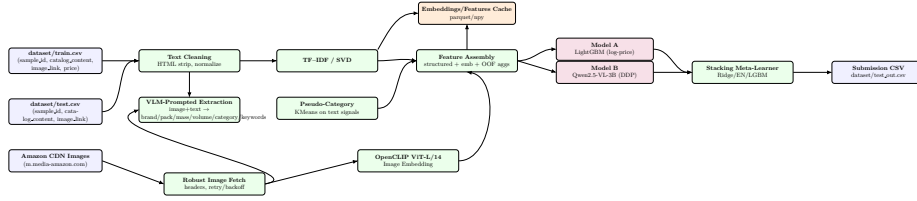


Figure 1: Clean engineering diagram for the multimodal pricing pipeline without overlaps, using curved routing and consistent node spacing.