Multimodal Product Price Prediction Pipeline

Technical Documentation

Amazon ML Challenge 2024

1 Overview

This document describes a comprehensive machine learning pipeline for predicting product prices using multimodal data (text descriptions and product images). The design draws inspiration from previously successful solutions (notably the 2023 winning approach) while introducing enhancements and more modular embedding strategies.

2 Related Work / Acknowledgement

Our methodology is inspired by the **Amazon ML Challenge 2023** winner's solution, available at `https://github.com/pj-mathematician/Amazon-ML-Challenge-2023/blob/main/amazon-ml-challenge-2023-winner-solution.ipynb`. We extend their ideas of embedding-based feature engineering, KNN-derived statistics, and ensemble blending, adapting them to a multimodal setting with newer embedding architectures.

3 Problem Statement

Given product catalog information (text metadata) and product images, the task is to predict product prices. Performance is evaluated using the *Symmetric Mean Absolute Percentage Error (SMAPE)* metric.

4 Data Preprocessing

4.1 Target Transformation

We operate on **log-transformed prices** instead of raw prices to stabilize variance, normalize distributions, and emphasize relative differences.

Listing 1: Log Transformation of Target Variable

```
log_price = np.log(price)
```

Advantages:

- Stabilizes variance across price ranges

- Produces a more symmetric, near-normal distribution
- Mitigates impact of extreme outliers
- Guides the model to learn relative differences rather than absolute ones

5 Embedding Generation

5.1 Fine-Tuning with Triplet Loss

We fine-tune embedding models using **triplet loss** to obtain price-aware latent spaces. The learning objective encourages embeddings of similar-priced products to be close and dissimilar-priced products to be far apart.

5.1.1 Loss Definition

$$L(A, P, N) = \max\big(\|f(A) - f(P)\|_2^2 \; - \; \|f(A) - f(N)\|_2^2 + \alpha, \; 0\big)$$

where:

- A: Anchor product
- P: Positive (similar-priced) product
- N: Negative (dissimilar-priced) product
- $f(\cdot)$: Embedding function (neural encoder)
- α: Margin enforcing separation

5.1.2 Triplet Sampling Strategy

- **Positive examples**: Products whose prices lie within $\pm 10\%$ of the anchor
- **Negative examples**: Products whose prices differ by more than 30% from the anchor

This encourages the model to minimize distance between similarly priced items and enforce a margin relative to more distant-priced items.

5.2 Models Fine-Tuned with Triplet Loss

We fine-tune multiple embeddings:

- CLIP Text Encoder
- ViT (Vision Transformer)
- DistilBERT (Single Field)
- DistilBERT (2-Field Concatenated)

5.3 Structured Embeddings

Beyond the fine-tuned models, structured embeddings are constructed leveraging pre-trained models:

- Title embeddings (512 dimensions)
- Description embeddings (512 dimensions)
- Category/attribute embeddings (variable dimension)

These embeddings are concatenated or combined to form a unified feature vector per product.

6 KNN Feature Construction

For each embedding model, K-nearest neighbors (via cosine similarity, implemented through FAISS) are identified. We derive statistical features from neighbor prices:

- Mean, median, min, max
- Standard deviation
- Inverse-distance-weighted average
- Quantiles (25th, 75th)

This is repeated for all embedding models (fine-tuned + non-fine-tuned), capturing multiple local views of price structure.

7 Meta-Learner Architecture

7.1 Model Choice

We use **CatBoost** as the primary meta-learner, due to its robustness to mixed data types and strong default performance.

7.2 Training Strategy

- **K-Fold Cross-Validation:** for reliable generalization
- **Out-of-Fold Predictions:** used to prevent overfitting in blending
- **Early Stopping:** monitors SMAPE on validation folds

8 Feature Importance Interpretability

We compute **SHAP values** to identify feature contributions:

- Fine-tuned embeddings (CLIP, ViT, DistilBERT) rank high in importance
- Pretrained/non-fine-tuned embeddings (e.g. SigLIP, Nomic, Florence-2) supply complementary signals, notably in less-represented categories

9 Pipeline Workflow

9.1 Stage 1: KNN Feature Extraction

- Use FAISS to find K nearest neighbors per embedding
- Extract statistical summaries for each embedding
- Combine into a unified feature matrix

9.2 Stage 2: Initial LightGBM Training

Train a LightGBM model on the full feature set to get preliminary predictions and feature importance.

9.3 Stage 3: Feature Filtering & Dual Training

Filter out weak or redundant features based on SHAP importance. Then train two models on the filtered feature set:

- CatBoost
- LightGBM

Generate out-of-fold predictions for blending.

9.4 Stage 4: Multi-K Blending

Blend final predictions from models trained with different neighbor sizes:

- $K = 5$: emphasizes local, fine-grained price patterns
- $K = 10$: balances local and global patterns, adds stability

10 Conclusion

This pipeline synthesizes approaches from prior successful solutions (especially the 2023 winner) with new innovations in multimodal embedding and feature engineering. The combination of fine-tuned and pretrained embeddings, KNN-derived statistics, and ensemble meta-learning offers both flexibility and interpretability in price prediction.