Amazon ML Challenge 2025

The Learning Rate

Abstract

We frame smart product pricing as supervised regression from product text (title/description/IPQ) and optional tabular signals to a strictly positive price. The approach fine-tunes a pretrained Transformer text encoder, fuses learned text embeddings with auto-discovered numeric features, trains a lightweight MLP head on log-mapped prices, and evaluates with SMAPE. The pipeline is leakage-aware, reproducible, and submission-ready.

1 Problem Framing & Metric

Given rows with an identifier i (e.g. `sample_id`), free-text `catalog_content`, optional tabular columns, and (for train) a target price $y_i > 0$, learn $f(\cdot)$ to predict $\hat{y}_i > 0$ on test rows. The leaderboard metric is Symmetric Mean Absolute Percentage Error (SMAPE):

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

with a small ϵ (e.g. 10^{-9}) for numerical stability. We are required to output a CSV with columns `sample_id,price` for every test row and ensure prices are strictly positive.

2 High-Level Design

Backbone. A pretrained Transformer encoder (e.g. DistilBERT/DeBERTa) tokenizes `catalog_content` and produces contextual embeddings. We mean-pool the last hidden states with the attention mask to obtain a fixed-size text vector $h_i^{\text{text}} \in \mathbb{R}^d$.

Tabular fusion. We automatically discover numeric/boolean columns (excluding ID, text, label, and any leakage-prone fields) and standardize them to produce $h_i^{\text{tab}} \in \mathbb{R}^p$. The fused representation is $h_i = [h_i^{\text{text}}; h_i^{\text{tab}}]$.

Head & link. A two-layer MLP with dropout maps h_i to a scalar z_i trained against $\log(1 + y_i)$. At inference we apply $\hat{y}_i = \max\{\exp(z_i) - 1,\ \varepsilon\}$ with $\varepsilon \approx 10^{-6}$ to satisfy positivity.

3 Data Pipeline & Leakage Control

Columns. We expect: `sample_id` (ID), `catalog_content` (text), and `price` (train only). Missing text is filled with an empty string for tokenizer consistency.

Auto tabular discovery. From all CSV columns we select numeric/boolean features and exclude: (i) `sample_id`, `catalog_content`, `price`; (ii) any column matching a leakage-safe pattern such as `/(^y$|price_per|price|log_price)/i`.

Scaling. For the selected tabular set, compute per-column mean and std on the training fold, impute NaNs with the training mean, and standardize. Persist both the exact column list and scaler statistics to ensure identical test-time preprocessing.

4 Model Architecture

Text encoder. HuggingFace `AutoModel` with max sequence length (e.g. 256). We avoid passing `token_type_ids` to encoders that do not accept them (e.g. DistilBERT).

Pooling. Let $H \in \mathbb{R}^{L\times d}$ be last-layer token states and $m \in \{0,1\}^L$ the attention mask. Mean-pooling:

$$h^{\text{text}} \;=\; \frac{\sum_{t=1}^{L} m_t\, H_t}{\sum_{t=1}^{L} m_t}.$$

Fusion head. With $h = [h^{\text{text}}; h^{\text{tab}}]$, the MLP computes

$$z \;=\; W_2\,\sigma(\text{Dropout}(W_1 h + b_1)) + b_2, \quad \hat{y} \;=\; \max\{\exp(z) - 1, \varepsilon\},$$

and we minimize MSE on $\log(1+y)$.

5 Training, Validation & Optimization

Splits. Single holdout or K-Fold CV. For each fold: (1) fit the tabular scaler on the training split, (2) fine-tune the model with early stopping, (3) save checkpoint and tabular config. Test predictions are averaged across folds.

Loss/metric. Train with MSE on $\log(1+y)$; during evaluation, transform predictions back to price space and compute SMAPE, MAE, and RMSE.

Optimization. `AdamW` with weight decay, warmup ratio, gradient accumulation as needed, and optional `fp16`. Early stopping monitors validation SMAPE (lower is better).

6 Inference & Submission

For each trained fold, build the test dataset using the saved tabular column list and scaler stats from that fold. Average fold predictions per test row:

$$\hat{y}_i^{\text{final}} \;=\; \frac{1}{K}\sum_{k=1}^{K} \hat{y}_i^{(k)}.$$

Emit `test_out.csv` with exact columns `sample_id,price` in the same order as the test CSV.

7 Reproducibility & Modularity

Seeds. Set a global `random_state`; optionally offset by fold index for diverse but reproducible ensembling.

Artifacts. Persist for each fold: model weights, tokenizer, and `tabular_config.json` (column list, means, stds). This guarantees deterministic inference across machines.

CLI design. Flags mirror common HF training scripts to simplify hyperparameter sweeps and scheduler integration.

8 Recommended Defaults & Tips

- **Backbone:** `distilbert-base-uncased` (swap to `microsoft/deberta-v3-base` for stronger baselines).
- **Seq length:** 256 (increase to 320–384 for longer descriptions).
- **Batch/LR/Epochs:** 16 / 2×10^{-5} / 3; weight decay 0.01; warmup ratio 0.06.

- **Folds:** 1 for speed; 5 for stability (average predictions).
- **Tabular mode:** auto-discovery with leakage-safe exclusion.
- **Precision:** enable `--fp16` on supported GPUs.

9 Limitations & Risk Mitigation

No image pixels. The baseline ignores image modalities; add a vision encoder for late fusion if images contain pricing cues.

Distribution shift. Style/domain drift in descriptions may raise SMAPE; mitigate via CV, regularization, and robust feature curation.

Truncation. Very long texts may be truncated at the chosen max length; consider increasing length or lightweight summarization.

10 How to Run

Quick run (single split):

```
python train_regression_transformer_tabular.py \
  --train_csv dataset/train.csv \
  --test_csv dataset/test.csv \
  --output_csv test_out.csv \
  --output_dir checkpoints \
  --model_name distilbert-base-uncased \
  --folds 1 --valid_size 0.1 --fp16
```

5-Fold CV with auto tabular features:

```
python train_regression_transformer_tabular.py \
  --train_csv dataset/train.csv \
  --test_csv dataset/test.csv \
  --output_csv test_out.csv \
  --output_dir checkpoints \
  --model_name microsoft/deberta-v3-base \
  --folds 5 --patience 5 --log_every_steps 200 --fp16
```

11 Summary

We fine-tune a Transformer on product text, fuse optional standardized numeric features, train in log space for stability and positivity, validate with SMAPE on the original scale, and ensemble across folds. The pipeline is leakage-aware, reproducible, and produces a compliant submission.

Artifacts produced: best model per fold, tokenizer, scaler config, and `test_out.csv`.