Multimodal Ensemble Model for Product Price Prediction

Group members:

1) N. Divyagnan Reddy – 2201130
2) V.Pranay Satvik Reddy – 2201221
3) T. Satya Pranav – 2201213
4) V. Sriniketh Reddy - 2201226

The Problem & Objective

- **The Goal:** To accurately predict the price of a given product using its catalog information.
- **The Challenge:** A product's price is determined by a complex combination of factors:
- **Textual Data:** What is it? (e.g., "Organic", "Pack of 6", "Gluten-Free")
- **Visual Data:** What does it look like? (e.g., Premium packaging, bulk item, single item)
- **Quantitative Data:** What is its size? (e.g., 12 oz, 1.5 L)
- **Our Objective:** Build a robust machine learning model that leverages all these modalities to produce a single, reliable price prediction.

Data Overview & Preprocessing

- **1. Input Data Sources:**
- **`train.csv`:** Provided product catalog content (Item Name, Description) and the target `price`.
- **`train_embeddings.parquet`:** Provided pre-computed image embeddings. We used the **`clip_embedding`** (a 512-dimension vector from ViT-B/32).
- **2. Target Transformation (Critical Step):**
- Product prices are heavily right-skewed (many cheap items, few expensive ones).
- To fix this, we trained the model to predict the **log-transformed price**: `target = np.log1p(price)`
- All final predictions must be reversed using `price = np.expm1(log_prediction)`.

Methodology: Two-Model Ensemble

- solution was to build two "specialist" models and combine their predictions. No single model has all the answers.
- **Model 1: The "Text Expert"**
- **Analyzes:** The product's text (`item_name`, `desc`) and its structured data (`pack_size`, `brand`).
- **Technology:** TF-IDF + LightGBM
- **Model 2: The "Image Expert"**
- **Analyzes:** The product's visual features (`clip_embedding`) and its structured data.
- **Technology:** CLIP Embeddings + LightGBM
- **Final Prediction:** `Final Price = (Prediction_Text + Prediction_Image) / 2`

Feature Engineering

- **Text Processing:**
  - Parsed `item_name` and `desc` from the raw catalog.
  - Created `full_text_cleaned` by lowercasing, removing special characters, and stripping measurement units (e.g., "oz", "pack of 6") to avoid data leakage.
- **Numerical Features:**
  - `pack_size`: Extracted from item name (e.g., "Pack of 6" -> 6).
  - `parsed_value`: The numerical value from the form (e.g., 12 for "12 oz").
  - `total_volume`: Set to `parsed_value` (simplified from the training script).
  - `item_name_length`, `desc_length`, `has_description`.
  - **Log Transforms:** Created `log_pack_size` and `log_total_volume` to stabilize variance.
- **Categorical Features:**
  - `brand`: Extracted from the first word of the `item_name`.

Model 1 Architecture: Text based

- **Pipeline:** `text_model.pkl`
- This model is a scikit-learn pipeline that combines feature processing and a regressor.
- **ColumnTransformer (Preprocessor):**
  - **Numerical Features:** `SimpleImputer` (median) -> `RobustScaler`.
  - **Categorical (`brand`):** `SimpleImputer` (constant) -> `OneHotEncoder`.
  - **Text (`full_text_cleaned`):** `TfidfVectorizer` (max 5000 features, 1-2 ngrams).
- **Regressor:**
  - `LGBMRegressor` (LightGBM)
  - Trained on the combined output of the preprocessor.
  - **This model learns patterns like "organic" or "pack of 12" from the text.**

Model 2 Architecture: Image based

- **Pipeline:** `image_model.pkl`
- This model's architecture is similar, but it swaps the TF-IDF text features for the image's CLIP features.
- **ColumnTransformer (Preprocessor):**
  - **Numerical Features:** `SimpleImputer` (median) -> `RobustScaler`.
  - **Categorical (`brand`):** `SimpleImputer` (constant) -> `OneHotEncoder`.
  - **Image (`clip_0...clip_511`):** The 512 CLIP features are passed through a `StandardScaler`.
- **Regressor:**
  - `LGBMRegressor` (LightGBM)
  - Trained on the combined output of this preprocessor.
  - **This model learns visual patterns, like the difference between premium packaging and a generic brand.**

Inference & API Architecture

- The backend API (FastAPI) is built to perfectly replicate the training process for live predictions.
- **Startup:** Loads `text_model.pkl`, `image_model.pkl`, and the raw `CLIP` model into memory.
- **Request:** Receives `item_name`, `description`, `value`, and an `image`.
- **Feature Engineering:**
  - Runs the *exact* same Python functions (e.g., `extract_pack_size`, `clean_text`) on the input data.
  - Uses the raw `CLIP` model to generate a 512-dim embedding for the new image.
- **Prediction:**
  - Creates a single-row `DataFrame` containing all 500+ features.
  - `log_price_text = text_model.predict(df)`
  - `log_price_image = image_model.predict(df)`

Reported Results

Performance on 20% Hold-Out Test Set
Fill this table with the final SMAPE and MAE values printed by your training script.
Key Insights:

- The ensemble model (simple average) performed better than either specialist model alone.
- This proves that **both text and image data contain unique, valuable signals** for price prediction.
- Aligning feature engineering between training and inference was the key to fixing initial prediction errors (like $0.00).

Model	SMAPE (Lower is Better)	MAE (Lower is Better)
Model 1 (Text Expert)	26.8 %	$ 12.14
Model 2 (Image Expert)	27.85 %	$ 12.78
Final Ensemble (Average)	**26.6 %**	**$ 12.14**

Conclusion

- **Conclusion:** We successfully built a multimodal price prediction engine by ensembling two specialist LightGBM models. This hybrid approach is robust and leverages the strengths of both text (TF-IDF) and image (CLIP) data, outperforming either modality on its own.
- **Future Improvements :**
- **Stacked Ensemble:** Instead of a simple average, train a "meta-model" (e.g., a `Ridge` regression) on the predictions of the two models to learn the optimal blend.
- **Hyperparameter Tuning:** Systematically tune the `LGBMRegressor` parameters (e.g., `n_estimators`, `learning_rate`) for both models to improve accuracy.
- **Richer Features:**
  - Use Optical Character Recognition (OCR) to extract text directly from the product images.
  - Use the `resnet_embedding` in addition to the `clip_embedding`.
  - Fine-tune a dedicated multimodal transformer model on this dataset.

Thank you