Mapply AI Assessment Summary

Name: Shreyas Waikar
Final Setup: K = 5, Top-N = 2
Embedding Model: all-MiniLM-L6-v2 (Sentence Transformers)

Objective

The goal of this assessment was to build a lightweight product classification system that can categorize Amazon product titles into relevant categories using vector embeddings and nearest-neighbor search.

My Approach

I started by using the bprateek/amazon_product_description dataset from HuggingFace. I focused on product_title and split the multi-level category field into individual tags to better support multi-label classification. The data was then split into training and test sets (80/20), ensuring balanced representation by stratifying on the primary tag.

For embeddings, I used the pretrained all-MiniLM-L6-v2 model. The text input for embeddings combined product_title with about_product for additional context. These embeddings were indexed using FAISS, and I experimented with both IndexFlatL2 and IVFFlat.

During classification, I tested various values of k (3, 5, 8) and different Top-N tag outputs (Top-1, Top-2, Top-3). Predictions were generated by aggregating tags from the nearest neighbors and selecting the most frequent ones.

Key Metrics

Setting	Accuracy	Precision	Recall	F1
Top-2, K=5	0.7083	0.8067	0.4802	0.5895
Top-1	0.8877	0.8877	0.2692	0.4064
Top-3	0.5322	0.7308	0.6341	0.6649
Top-2, K=3	0.6957	0.7950	0.4739	0.5813
Top-2, K=8	0.6985	0.8059	0.4810	0.5897
Top-2, with mpnet	0.6968	0.8010	0.4793	0.5870

Final Decision

After evaluating multiple combinations, I selected Top-2 predictions with K=5 using all-MiniLM-L6-v2. This setup offered the best balance between high precision and reasonable recall. Top-1 was very precise but too conservative, and Top-3 increased recall but diluted clarity.

Insights

To better visualize model performance, I created a confusion matrix focused on the Top-20 most frequent tags. Including numeric values made it easier to spot common misclassifications and understand model behavior at a glance.

Future Improvements

If I had more time, I would explore:

- Using richer features like product_description, brand, or technical details
- Fine-tuning the embedding model for this domain
- Training a supervised classifier on top of the embeddings for improved label separation