Qogita challenge report – Kamesh Kotwani

Introduction

The purpose of this project is to build a machine learning model that predicts a product's category based on partial information, namely the product title, brand name, and image. This addresses a key challenge in building structured product catalogues from inconsistent supplier data which is a common scenario in large-scale B2B marketplaces.

Aim

- Predict the product's category using product title, brand name and image.
- Build an end-to-end ML pipeline from preprocessing to deployment.
- Serve the model via a FastAPI endpoint, containerised with Docker.

Glossary of Terms

Terms	**Description**
CatBoost	Gradient boosting library designed for categorical features
LabelEncoder	Encodes target/class labels as integers
MLflow	Open-source platform for experiment tracking
API	Application Programming Interface used for serving model predictions
PyTorch	Library for deep learning used for computer vision and natural language processing.
PCA (Principal Component Analysis)	Dimensionality reduction technique used on image features
(TF – IDF) Term Frequency–Inverse Document Frequency	A text vectorisation method that weighs words based on uniqueness

Data Definition

Feature	Type	Description
name	Text	Title of the product
brandName	Categorical	Name of the supplier brand
imageUrl	Visual	URL to product image
categoryName	Categorical	Target variable (product category)

Methodology

Preprocessing

- Text columns (name and brandName) were preprocessed using **NLTK**, including:
  - Lowercasing
  - Tokenisation using word_tokenize
  - Stopword removal (stopwords.words("english"))
  - Lemmatisation with WordNetLemmatizer
- The processed text was retained for use with CatBoost's text_features capability, which supports raw string inputs internally.

Image Feature Extraction

- Product images were downloaded and processed through a **pretrained ResNet50** model (from torchvision):
  - Final layer removed to extract 2048-dimensional feature vectors
  - Features were reduced to 100 dimensions using **PCA** (n_components=100)
  - Image vectors were appended as **numerical features** to the tabular dataset

Feature Engineering

- name was passed to CatBoost as a **text feature**
- brandName was passed as a **categorical feature**
- ResNet + PCA-reduced image vectors were passed as **numerical features**
- The full feature set was assembled and split into training and testing sets

Model Training with textual data

- **CatBoostClassifier** was used to train across 1160 training rows.
- Experiments were tracked using **MLflow**, with:

- Project name: Catboost Vanilla
- Centralised tracking via MLFLOW_TRACKING_URI

Model Training with textual and image data

- The model used was **CatBoostClassifier**, chosen for its:
  - Native support for **categorical** and **text** features
  - Compatibility with **numerical** image vectors
  - Built-in handling of missing values and unknown categories
- Model training was performed with:
  - loss_function='MultiClass' for multi-class prediction
  - GPU acceleration (task_type='GPU')
  - early_stopping_rounds=50 for efficiency

Hyperparameter Search

- A parameter grid was defined and explored using ParameterGrid:
  - iterations: [300, 500, 1000]
  - depth: [6, 7]
- For each combination:
  - A **nested MLflow run** was created
  - The model was trained using GPU (task_type="GPU")
  - **Early stopping** was applied with early_stopping_rounds=20

Results and Conclusions

Model Performance Summary

Configuration	Accuracy
Textual Data Only	0.739
Text + Image Features	0.740

When training the model using only **textual features** (name and brandName), the CatBoost classifier achieved an accuracy of approximately **73.9%.** After incorporating **image embeddings** extracted via ResNet50 (reduced to 100 dimensions using PCA), the accuracy improved marginally to **74.0%**.

However, looking beyond accuracy:

- The **macro average F1-score** was **~0.33**, indicating that the model struggled with underrepresented or minority classes.
- The **weighted F1-score** was **~0.69**, showing that the model performed better on more frequent categories. (Eau De Parfum being the highest category).

- Class imbalance significantly impacted the ability to generalise across all product categories.

Strengths of the Approach

- **Multi-modal pipeline**: The model integrates text, categorical, and visual data into a single, production-ready pipeline.
- **Minimal preprocessing required**: Leveraged CatBoost's native support for both text and categorical features, simplifying feature engineering.
- **Efficient experimentation**: MLflow was used throughout to track experiments, parameters, and metrics, making comparisons between runs seamless.
- **Scalable setup**: The model was deployed via FastAPI and Docker, ensuring reproducibility and easy serving of predictions.

Weaknesses and Limitations

- **Minimal gain from image features**: Despite extracting deep features from product images, their impact on performance was limited. This may be due to:
    - Overlap in visual features across classes
    - No pre-trained model for specific cosmetic use-case. (A good opportunity to develop our own)
- **Extreme class imbalance**: Several categories had very few samples, limiting the model's ability to generalise well on rare classes.
- **Un-Stratified Sampling**: Due to extreme rare classes, stratified sampling didn't work since it requires at least 2 rows for each class, which was not the case.
- **Lack of contextual understanding in text**: TF-IDF and token-level text processing do not capture semantic relationships or context.

Future Improvements

- **Advanced Multi-Modal Architectures:**
    - Explore end-to-end networks that jointly learn from images and text.
    - Fine-tune multi-modal transformer models (e.g., VisualBERT, ViLBERT, LXMERT).
    - Leverage pretrained multi-modal models like CLIP for domain-specific fine-tuning.
- **Enhanced Text Embedding Strategies:**
    - Fine-tune domain-specific language models on product descriptions.
    - Apply contextual data augmentation (e.g., back-translation, synonym replacement).

    - Experiment with transformer-based embeddings (e.g., BERT, Sentence Transformers).
- **Improved Image Feature Extraction:**
    - Fine-tune CNN backbones (e.g., ResNet, EfficientNet) on cosmetics images.
    - Use deeper layers or intermediate feature maps for richer embeddings.
    - Consider nonlinear dimensionality reduction methods (e.g., autoencoders) instead of only PCA.
- **Handling Class Imbalance More Effectively:**
    - Use dynamic loss functions such as focal loss to emphasize minority classes.
    - Apply advanced resampling techniques (e.g., synthetic data generation via conditional GANs).
    - Explore cost-sensitive ensemble approaches to better balance predictions.
- **Hyperparameter & Architecture Optimization:**
    - Employ Bayesian optimization or Neural Architecture Search (NAS) for efficient tuning.
    - Utilize model interpretability tools (e.g., SHAP, LIME) for targeted feature selection.
- **Incorporating Additional Data Sources:**
    - Enrich the dataset with product metadata (ingredient lists, user reviews, pricing).
    - Integrate user interaction data (click-through, conversion rates) for more robust training.
- **Robust Evaluation & Continuous Improvement:**
    - Implement stratified cross-validation or time-series splits for reliable validation.
    - Continuously analyze misclassifications to identify new avenues for improvement.