Fashion Product Recommendation System — Simple Explanation Document

1. Project Goal

We want to build a **fashion product recommendation system** that can show:

- **Visually similar products**
- **Stylistically similar products**
- Or a **combination of both**

using a dataset that contains:

- **44k product JSON files** (metadata)
- **44k corresponding images**

🧩 2. Data Understanding

Each item has two types of data:

A) Metadata (from JSON files)

Examples:

- brand
- gender
- age group
- season
- colour
- price / discounted price
- product type
- pattern
- fabric
- returnability
- inventory availability

This data tells us **what the item is**, not how it looks.

B) Product Images

These contain **visual style**, like:

- colour shade
- silhouette
- texture
- pattern
- shape

The image is often the *strongest* signal in fashion recommendation.

🛠 3. Data Parsing & Cleaning

✔ Parsed 44k JSON files → extracted "data" section only

We wrote a JSON parsing function to extract important fields and convert them into a DataFrame.

✔ Cleaned null values, inconsistent fields, duplicated records

- Removed unnecessary columns
- Handled missing values
- Converted price, rating, vat to numeric
- Merged certain categorical features

✔ Removed features with >90% missing values

(This keeps the dataset clean and prevents noise.)

🧪 4. Feature Engineering

This is the MOST important step on the metadata side.

We created meaningful features that better represent fashion style.

Key engineered features:

(1) discount_ratio

discount_ratio = discount_amount / price

Helps understand discount impact.

(2) is_discounted

Binary feature indicating discount presence.

(3) log_price and log_effective_price

Log transform helps stabilize price variation.

(4) price_bucket

Categorizes products into:

- low
- mid
- high

Useful for similarity matching.

---

(5) style_signature (VERY IMPORTANT)

A combined text description made from:

- gender
- subCategory
- articleType
- baseColour
- pattern
- fabric
- occasion

Purpose:
Allows TF-IDF to capture fashion semantics like:

“women floral summer dress casual cotton”

---

(6) season_group

Grouped seasons into:

- warm
- cold
- all-year

(7) segment

Combines gender + ageGroup.

(8) category_strength

Counts how common each articleType is.

🖼️ 5. Image Feature Extraction (Visual Embeddings)

To understand image style, we used:

⭐ CLIP (ViT-B/32, LAION pretrained)

A state-of-the-art model trained on **400M+ image–text pairs**.

CLIP converts each image into a **512-dimensional embedding** that captures:

- colour
- pattern
- shape
- texture
- overall visual style

✔ We computed embeddings for all 44k images

Stored them in:

- image_embeddings_clip.npy
- image_embeddings_ids.csv

🧠 6. Creating the Feature Matrices

We created two types of feature matrices:

A) X_meta_text (Metadata + Text)

This includes:

1. Scaled numerical features

(price, vat, rating, discount ratio, etc.)

2. One-hot encoded categorical features

(brand, gender, colour, season, articleType, etc.)

3. TF-IDF representation of style_signature

We reduce this using SVD (64 dims).

🔶 **Why?**

This gives a **semantically rich representation** of fashion items based on metadata.

B) Image Embeddings

512-dim vectors from CLIP.

These represent **visual similarity**.

🔗 7. Three Types of Recommenders

1Image-Based Recommender

Uses **cosine similarity** on CLIP embeddings.

Returns products that **look visually similar**.

---

2 Metadata/Text Recommender

Uses **cosine similarity** on X_meta_text.

Returns products that are **similar in style**, considering:

- price
- colour
- season
- fashion attributes
- usage
- pattern
- category

---

3 Hybrid Recommender

Combines both similarity scores:

hybrid = α * image_sim + (1 - α) * meta_sim

Example:

α = 0.6   (image slightly more important)

This gives the **best real-world recommendations**.

---

8. Clustering (Optional but Useful)

We applied **MiniBatch KMeans** on X_meta_text to create ~30 clusters.

Purpose:

- Group products into "style families"
- e.g., cluster 5 = "women summer skirts"
- Improves recommendation quality
- Useful for visualization & reporting

But clustering is **not required** for the core recommender.

---

9. Dimensionality Reduction for Visualization

We reduced the vectors to **2D** using SVD to plot clusters.

This allows us to show:

- How items are grouped
- How train vs test data fall into clusters

Good for presentation.

🏁 10. Final Outputs

Your system can now:

✔ **Retrieve visually similar products**

✔ **Retrieve stylistically similar products (metadata/text)**

✔ **Combine both for hybrid recommendations**

✔ **Organize products into meaningful style clusters**

✔ **Visualize the embedding space**

This is exactly how modern fashion recommender systems work.

🌟 11. Why This Approach is Correct

Because:

- We **don't have user history** → cannot use collaborative filtering
- Fashion similarity is fundamentally based on:
  - **visual style**
  - **fashion attributes**
  - **product metadata**

CLIP + TF-IDF + engineered features + cosine similarity
is the industry-standard approach for fashion-product retrieval.
