1. Ce qui peut définir le « comportement utilisateur » dans ton jeu de données

Tu peux voir ton dataset comme une mine de signaux comportementaux. Voici ce que tu pourrais en extraire pour chaque utilisateur :

1.1 Volume d'interactions & récence

Tu pourrais par exemple calculer :

- Le **nombre de produits** avec lesquels l'utilisateur a interagi (achetés / évalués).
- La **fréquence de reviews** (par mois, par trimestre…).
- Des **features de récence** : temps depuis la dernière interaction, nombre d'actions sur les X derniers jours/mois.
- Des **patterns de saisonnalité** : plus actif à certaines périodes (ex. pics de cadeaux en décembre).

1.2 Préférences de catégories & sous-catégories

À partir de `main_category`, `categories`, etc., tu pourrais :

- Calculer la **répartition des interactions par catégorie** (ex. 60 % Beauty, 20 % Home, 20 % Electronics).
- Affiner avec des **sous-catégories** (ex. "Nail Art", "Hair Care").
- Mesurer l'**entropie** de cette distribution pour voir si un user est plutôt spécialiste (focus sur une catégorie) ou généraliste.

1.3 Sensibilité au prix & profil de dépense

Avec la variable `price`, tu peux :

- Calculer un **prix moyen / médian** des produits achetés.
- Mesurer la **variance de prix** (est-ce qu'il varie entre produits très cheap et très chers ?).
- Créer les **bandes de prix** (bas / moyen / élevé) et regarder la proportion dans chaque bande.
- Construire des features de type **RFM** :
    - **R**ecency : depuis quand n'a-t-il rien acheté ?
    - **F**requency : combien d'achats sur une période donnée ?
    - **M**onetary : total dépensé, moyenne par achat.

1.4 Affinité marque / boutique

À partir de `Brand`, `store`, tu peux :

- Identifier les **marques les plus achetées** par user.

- Mesurer une **concentration de marque** (indice de Herfindahl, par exemple) → user fidèle à 1–2 marques vs explorateur.
- Faire la même chose au niveau des **stores**, si tu en as plusieurs.

1.5 Style de notation & d'avis

Sur la base de `rating` et `review_text`, tu pourrais :

- Calculer la **note moyenne** donnée (user exigeant vs généreux).
- Calculer la **variance des notes** (toujours entre 4–5, ou bien très variable ?).
- Regarder la **distribution des notes** (proportions de 1, 2, 3, 4, 5 étoiles).
- Extraire des infos du texte :
    - **Longueur moyenne** des reviews (en nombre de mots).
      Score de sentiment approximatif (positif / neutre / négatif).
    - Des **thèmes** via du topic modeling ou des embeddings (ex. l'utilisateur parle surtout de "durabilité", "odeur", "packaging").

1.7 Patterns temporels

Si tu as des timestamps, tu peux aussi :

- Regarder les **heures/jours d'activité** (plutôt soir, week-end…).
- Séparer **intérêts long terme vs court terme** :
    - Embedding global sur tout l'historique.
    - Embedding sur les **N dernières interactions** (ce qui reflète l'intention actuelle)

Tous ces éléments peuvent t'aider à :

- **Segmenter / clusteriser** les utilisateurs,
- Et construire de **meilleurs modèles de recommandation** (features d'entrée pour des modèles plus avancés).

2.1 Stratégies très simples / baselines

Je te conseille de commencer par quelques approches basiques, pour avoir des points de comparaison :

1. **Popularité globale**
    - Recommander les produits **les plus achetés / les plus notés** sur l'ensemble du site.
    - Variantes possibles :
        - Popularité **par catégorie** ("Top Beauty", "Top Nail Tips"…).
        - Popularité en filtrant sur la **qualité** (ex. note ≥ 4 et nombre de reviews ≥ Y).

2. **Produits “tendances” / récemment populaires**

    - Utiliser des **comptes pondérés par le temps** : les interactions récentes pèsent plus.
    - Tu peux construire des listes du type :
        - “Les plus populaires cette semaine dans Beauty”.
        - “Top des ventes du mois” vs “Top historique”.

3. **Nouveautés bien notées**
    - Si tu as une date de mise en ligne, tu peux recommander :
        - Les **nouveaux produits**,
        - Avec de **bons signaux précoces** (bonnes premières notes / reviews).

Ces baselines ne sont pas personnalisées, mais elles sont très utiles comme référence et pour les cas de **cold-start user**.

2 Recommandation par clustering

Une fois que tu as construit des features de comportement (section 1), tu peux aller vers le clustering.

(a) Clustering d’utilisateurs

Tu pourrais :

- Appliquer k-means, GMM, clustering hiérarchique, etc. sur des features telles que :
    - Distribution de catégories.
    - Bande de prix moyenne.
    - Concentration de marques.
    - Comportement de rating (moyenne, variance).
    - Vecteur de goût (embedding moyen des produits consommés).

Idée de recommandation :

- Pour chaque cluster, tu calcules la **popularité des produits au sein de ce cluster**.
- Pour un user donné :
    - Tu l’assignes à un cluster.
    - Tu lui recommandes les **Top-N produits les plus populaires** de son cluster (en excluant ceux qu’il connaît déjà).

Tu peux améliorer ça en :

- Mélangeant **popularité du cluster** et **popularité globale** (pour éviter les extrêmes).
- Combinant avec du CF plus fin :

- score(item, user) = α·popularité_cluster + (1−α)·score_personnalisé_CF.

(b) Clustering d'items

Tu peux aussi clusteriser les produits, par exemple avec :

- Le texte (embeddings de title + description + features).
- Les métadonnées (catégorie, brand, price, material, style).
- Les images (embeddings visuels).

Ensuite :

- Pour un produit donné, tu peux recommander des **items du même cluster** (similaires).
- Tu peux combiner ça avec des **co-occurrences** ("souvent achetés ensemble") pour des suggestions de type :
  - "Produits similaires"
  - "Les clientes qui ont vu ceci ont aussi consulté…"

3. Approche basée sur les métadonnées

Ici, l'idée est de t'appuyer uniquement sur les infos structurées du produit, sans forcément utiliser le texte ou l'historique.

À partir de `details` et autres champs, tu peux utiliser :

- `main_category`, `Brand`, `Material`, `Style`, `Price`, `dimensions`, etc.

Pour chaque produit, tu pourrais :

- Créer un vecteur :
  - **Catégoriel** : one-hot / multi-hot pour les catégories, marques, styles…
  - **Numérique** : prix, note moyenne, nombre de reviews, normalisés.
- Facultatif : ajouter d'autres features dérivées (log du prix, bin "low / medium / high price", etc.).

Pour l'utilisateur :

- Tu peux calculer la **moyenne (ou moyenne pondérée)** des vecteurs des produits qu'il a appréciés (par exemple ceux avec rating ≥ 4).
- Ce vecteur reflète un **profil préférentiel en termes de métadonnées** :
  - Plutôt telle catégorie.
  - Plutôt telle gamme de prix.
  - Plutôt certaines marques / matériaux / styles.

Ensuite, tu peux :

- Mesurer la **similarité** entre ce profil et les vecteurs des autres produits (cosine, dot product…).
- Recommander les produits dont le vecteur de métadonnées est le plus **proche** du profil de l'utilisateur.

C'est particulièrement utile pour :

- Les **nouveaux produits** (sans historique d'interactions).
- Les **utilisateurs avec peu d'historique**, si tu veux exploiter fortement les caractéristiques "objet".

4. Mise en place d'une évaluation offline

Pour évaluer tes modèles sans les déployer, tu peux organiser ton jeu de données de la façon suivante :

1. Split train / validation / test

Je te conseille un **split temporel** :

- Tu choisis une date T.
- Tu **trains** sur toutes les interactions *avant* T.
- Tu **tests** sur les interactions *après* T (c'est plus réaliste que de couper au hasard).

Tu peux aussi prévoir un petit **set de validation** (par ex. une période juste avant la période de test) pour tuner les hyperparamètres.

2. Stratégies de « leave-one-out » ou « leave-last-k-out »

Pour chaque utilisateur, tu peux :

- Garder ses **dernières interactions** comme **cible** (ex. la dernière ou les 3 dernières).
- Utiliser le reste de son historique comme **train**.
- Demander à ton modèle : "à partir de cet historique, aurais-tu réussi à prédire ces derniers items ?"