*Báo cáo BTL học phần này cần trình bày theo cấu trúc dưới đây, tuân thủ Quy định trình bày của khoa.*

# 

# XÂY DỰNG HỆ THỐNG GỢI Ý SẢN PHẨM KẾT HỢP PHÂN CỤM VÀ LỌC CỘNG TÁC DỰA TRÊN DỮ LIỆU NGƯỜI DÙNG

# CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

## 1.1. **Giới thiệu bài toán**

Trong hệ thống thương mại điện tử, việc đề xuất sản phẩm phù hợp cho người dùng không chỉ giúp tăng doanh thu mà còn nâng cao trải nghiệm mua sắm. Bài toán gợi ý sản phẩm dựa trên dữ liệu hành vi khách hàng hiện là một hướng nghiên cứu và ứng dụng quan trọng. Bên cạnh kỹ thuật khai phá luật kết hợp, các mô hình học máy đơn giản như **Collaborative filtering**, **KMeans** và **Random Forest** cũng có thể được áp dụng để phân loại hoặc dự đoán nhóm sản phẩm người dùng có khả năng mua tiếp theo, từ đó đưa ra chiến lược khuyến mãi phù hợp.

**1.2. Đánh giá đề tài liên quan**

Trong lĩnh vực hệ thống gợi ý, nhiều phương pháp đã được nghiên cứu để cá nhân hóa đề xuất sản phẩm cho người dùng. Tuy nhiên, các bài toán về gợi ý với người dùng mới (cold‑start) và đa dạng dữ liệu người dùng vẫn gây thách thức lớn. Dưới đây là tổng quan các hướng nghiên cứu tiêu biểu:

### 1.2.1. Collaborative Filtering (CF) cổ điển

Collaborative Filtering (User-based hoặc Item‑based) dựa vào dữ liệu tương tác lịch sử (rating, mua hàng, click, v.v.) giữa người dùng và sản phẩm. CF hoạt động hiệu quả khi lịch sử tương tác đầy đủ, nhưng gặp rất nhiều hạn chế với người dùng mới, bởi không thể xác định hành vi của họ nếu không có dữ liệu (Breese, Heckerman & Kadie, 1998)

### 1.2.2. Phân cụm người dùng bằng thuật toán KMeans

KMeans là một thuật toán học không giám sát dùng để phân nhóm người dùng theo hồ sơ cá nhân (tuổi, giới tính, thu nhập, thói quen chi tiêu). Nghiên cứu cho thấy rằng việc phân nhóm này là cơ sở hiệu quả để gợi ý sản phẩm phổ biến trong mỗi nhóm, đặc biệt hữu ích cho cold-start user (Panteli & Boutsinas, 2023)

### 1.2.3. Hệ thống gợi ý lai (Hybrid CF + KMeans)

Hướng tiếp cận hybrid kết hợp CF và clustering (như KMeans) đã được áp dụng nhiều trong nghiên cứu gần đây. Theo Erion Çano & Maurizio Morisio (2019), các mô hình lai giúp giải quyết hiệu quả vấn đề cold-start và dữ liệu thưa thớt (sparsity), đồng thời nâng cao chất lượng đề xuất so với CF đơn lẻ. Các công trình khác (Panteli & Boutsinas, 2023) cũng trình bày rõ quy trình: dùng clustering phát hiện nhóm người dùng rồi khai thác các mẫu hành vi phân biệt trong mỗi cụm để phục vụ gợi ý cho người dùng mới

Collaborative Filtering vẫn là phương pháp chủ đạo cho gợi ý khi có dữ liệu hành vi, nhưng gặp hạn chế với người dùng mới. KMeans rất phù hợp để xử lý cold-start bằng cách phân nhóm người dùng dựa trên hồ sơ. Mô hình hybrid kết hợp CF và KMeans đã được chứng minh trong nhiều nghiên cứu là vừa hiệu quả về độ chính xác, vừa giúp mở rộng phạm vi gợi ý – đặc biệt đối với người dùng mới và dữ liệu sparse.

Do đó, đề tài của bạn chọn hướng tiếp cận Hybrid CF + KMeans là hoàn toàn phù hợp với xu hướng học thuật hiện nay cũng như đáp ứng yêu cầu thực tiễn triển khai trong hệ thống gợi ý.

## 1.3. Mục đích đề tài

Mục đích của đề tài là xây dựng một hệ thống gợi ý mặt hàng yêu thích cho người dùng trong thương mại điện tử dựa trên đặc điểm cá nhân, như độ tuổi, giới tính, thu nhập, trình độ học vấn, v.v.

Đề tài sử dụng các thuật toán học máy đơn giản bao gồm KMeans, Collaborative Filtering và Random Forest để học mối liên hệ giữa hồ sơ người dùng và mặt hàng họ yêu thích, từ đó dự đoán loại sản phẩm phù hợp cho người dùng mới hoặc gợi ý sản phẩm cho chiến dịch tiếp thị cá nhân hóa.

## 1.4. Mục tiêu đề tài

Phân tích và xử lý dữ liệu hồ sơ người dùng, bao gồm các thuộc tính cá nhân như tuổi, giới tính, thu nhập, học vấn, tình trạng hôn nhân,...

Tiền xử lý dữ liệu: mã hóa nhãn (Label Encoding), xử lý thiếu dữ liệu, chuẩn hóa giá trị thuộc tính.

Huấn luyện và đánh giá mô hình phân loại các mặt hàng yêu thích dựa trên hồ sơ người dùng, sử dụng:

* KMeans
* Collaborative Filtering
* Random Forest Classifier

So sánh hiệu năng các mô hình bằng các chỉ số: Accuracy, Precision, Recall.

Trực quan hóa kết quả mô hình và xây dựng báo cáo giải thích mối liên hệ giữa các đặc điểm cá nhân và sở thích mua sắm.

Đề xuất chiến lược gợi ý mặt hàng theo từng nhóm người dùng, dựa trên phân tích đầu ra của mô hình.

## 1.5. Các ràng buộc kỹ thuật

#### Ngôn ngữ lập trình và công cụ sử dụng

* Ngôn ngữ chính: **Python 3.x**
* Các thư viện và công cụ bắt buộc:
* **pandas, numpy**: xử lý và thao tác dữ liệu.  
  **scikit-learn (sklearn)**: xây dựng và đánh giá mô hình học máy (Collaborative filtering , KMeans, Random Forest)
* **matplotlib, seaborn** hoặc **plotly**: trực quan hóa dữ liệu và kết quả mô hình.
* **Jupyter Notebook** hoặc **Google Colab**: môi trường lập trình chính.
* **Ràng buộc về dữ liệu:** Dữ liệu phải ở dạng hồ sơ người dùng, gồm các đặc trưng đầu vào như:
* Giới tính
* Tuổi
* Thu nhập
* Trình độ học vấn
* Nghề nghiệp
* Tình trạng hôn nhân
* Vị trí địa lý
* Thời gian truy cập, thời gian nghiên cứu sản phẩm,...
* **Nhãn mục tiêu (label) là:** Loại mặt hàng yêu thích (được mã hóa thành nhãn phân loại).
* **Dữ liệu phải được tiền xử lý đúng chuẩn, bao gồm:**
* Mã hóa biến phân loại (Label Encoding hoặc One-Hot Encoding)
* Chuẩn hóa dữ liệu số
* Xử lý giá trị thiếu nếu có
* Yêu cầu về mô hình học máy
* **Phải triển khai tối thiểu 3 mô hình học máy:**
* KMeans
* Random Forest Classifier
* Collaborative filtering
* **Cần thực hiện đầy đủ các bước:**
* Chia tập train/test (ví dụ 80/20)
* Huấn luyện mô hình
* Dự đoán và đánh giá
* **Các chỉ số đánh giá bắt buộc:**
* Accuracy, Precision, Recall, F1-score: Sử dụng cho Random Forest
* Chỉ số rmse, mae cho mô hình collaborative filtering
* Chỉ số silhouette score và davies bouldin score cho KMeans

## 1.6. Phân công và Kế hoạch thực hiện

### 1.6.1. Phân công nhiệm vụ

| **STT** | **Họ và tên** | **Nhiệm vụ được giao** |
| --- | --- | --- |
| 1 | Bùi Gia Bảo | - Xây dựng mô hình gợi ý hybrid (KMeans + CF)  - Triển khai mã nguồn chính  - Viết chương 4: Thử nghiệm và đánh giá |
| 2 | Nguyễn Phương Hải | - Phân tích và tiền xử lý dữ liệu  - Xây dựng mô hình Random Forest  - Viết chương 3: Thiết kế hệ thống |
| 3 | Nguyễn Thị Tươi | - Khảo sát các nghiên cứu liên quan  - Viết chương 1: Giới thiệu và tổng quan  - Hỗ trợ viết chương 2: Cơ sở lý thuyết |
| 4 | Nguyễn Hoàng Tuấn | - Cài đặt mô hình Collaborative Filtering  - Đánh giá mô hình bằng RMSE/MAE  - Viết phần phân tích kết quả CF |
| 5 | Nguyễn Văn Hoá | - Xây dựng mô hình KMeans, đánh giá bằng Silhouette và DBI  - Viết phần phân tích kết quả KMeans  - Tổng hợp, chỉnh sửa toàn văn báo cáo |

### 1.6.2. Kế hoạch thực hiện

| **Thời gian** | **Nội dung công việc chính** | **Người phụ trách** |
| --- | --- | --- |
| 15/7 - 16/7 | Thu thập và khảo sát dữ liệu; xác định hướng nghiên cứu | Cả nhóm |
| 16/7 - 18/7 | Tiền xử lý và phân tích dữ liệu | Hải, Tươi |
| 19/7 - 21/7 | Xây dựng các mô hình: Random Forest, CF, KMeans | Tuấn, Hoá, Bảo |
| 22/7 - 25/7 | Tích hợp mô hình hybrid; xử lý bài toán cold-start | Bảo, Hoá |
| 26/7 - 28/7 | Đánh giá, so sánh kết quả theo các chỉ số MAE, RMSE, v.v. | Tuấn, Bảo |
| 29/7 | Viết báo cáo, hoàn thiện luận văn | Tươi, Hải, Hoá |
| 30/7 - 5/8 | Chỉnh sửa, hoàn thiện lần cuối và nộp báo cáo | Cả nhóm |

# CHƯƠNG 2. TIỀN XỬ LÝ DỮ LIỆU

## 2.1. Tải dữ liệu từ Kaggle:

**import shutil**

**import os**

**path = kagglehub.dataset\_download("kartikeybartwal/ecommerce-product-recommendation-collaborative")**

**os.makedirs('data', exist\_ok=True)**

**for filename in os.listdir(path):**

**src\_file = os.path.join(path, filename)**

**dst\_file = os.path.join('data', filename)**

**if os.path.isfile(src\_file):**

**shutil.copy(src\_file, dst\_file)**

**print("Files have been copied to /data")**

## 2.2. Khám phá dữ liệu

### 2.2.1. Loại bỏ các cột thừa:

Trong dataset được tải từ Kaggle, cột ‘Unnamed: 0’ thực chất là index đã có sẵn trong dữ liệu ban đầu, nên sẽ bị loại bỏ.

**df = pd.read\_csv("data/user\_personalized\_features.csv")**

**df.head(10)**

**if 'Unnamed: 0' in df.columns:**

**df.drop(['Unnamed: 0'], inplace=True, axis=1)**

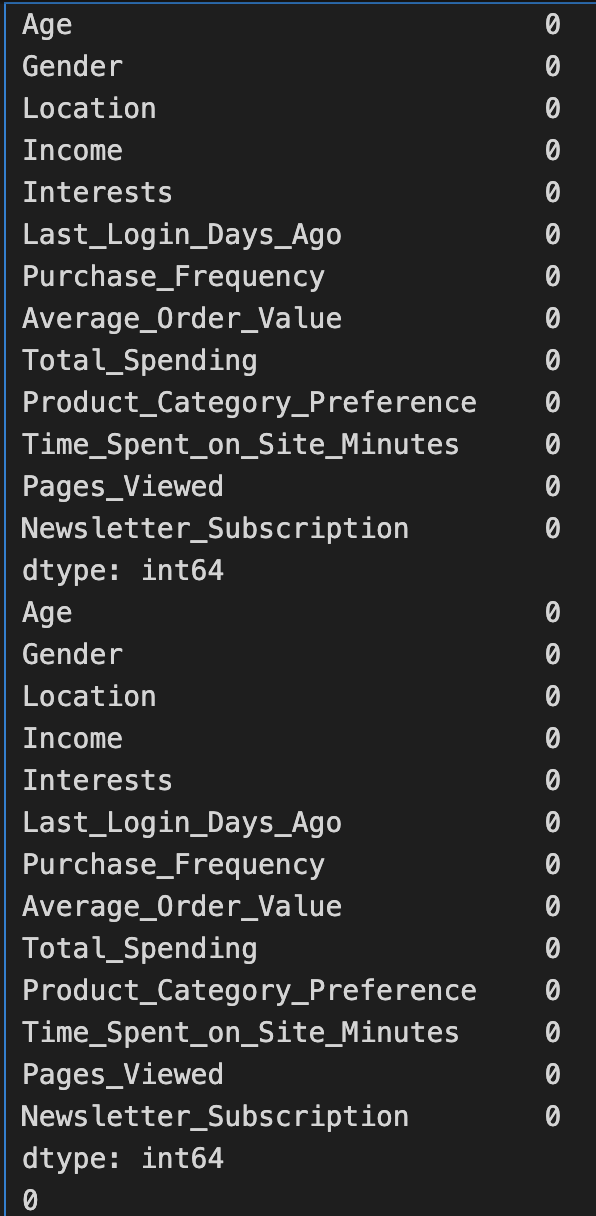
**df.head(10)**

### 2.2.2. Kiểm tra dữ liệu bị thiếu, bị lỗi

**print(df.isna().sum())**

**print(df.isnull().sum())**

**print(df.duplicated().sum())**

****

Dataset không có dữ liệu bị trùng lặp, cũng như dữ liệu null và nan, vậy nên có thể sử dụng luôn để phân tích và đào tạo mô hình.

### 2.2.3. Khám phá độ lệch của dữ liệu, phân bổ của dữ liệu và sự tương quan

Các biểu đồ cho thấy dữ liệu không có điểm ngoại lệ và điểm ngoại lai, nên không cần phải qua bước xử lý các điểm này. Ngoài ra, các dữ liệu được phân bổ tự nhiên.

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import numpy as np**

**df.hist(bins=20, figsize=(12, 6))**

**plt.tight\_layout()**

**plt.show()**

**cat\_cols = df.select\_dtypes(include=['object']).columns.to\_list()**

**num\_cols = df.select\_dtypes(include=['number']).columns.to\_list()**

**for col in cat\_cols:**

**plt.figure(figsize=(12, 6))**

**sns.countplot(data=df, x=col)**

**plt.title(f'Distribution of {col}')**

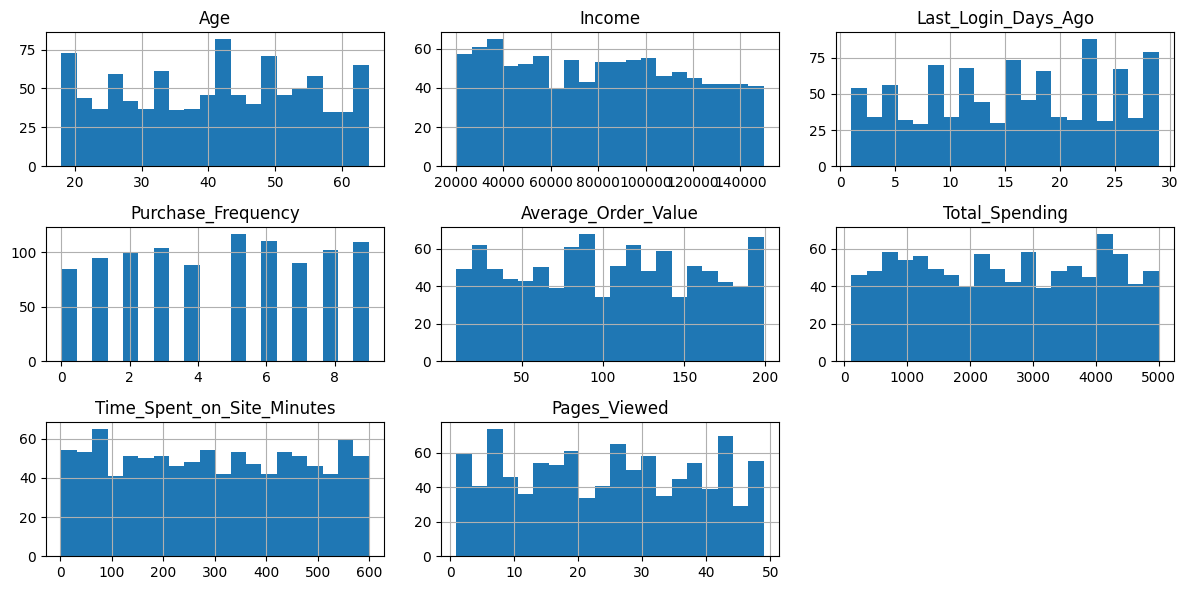
**plt.xticks(rotation=45)**

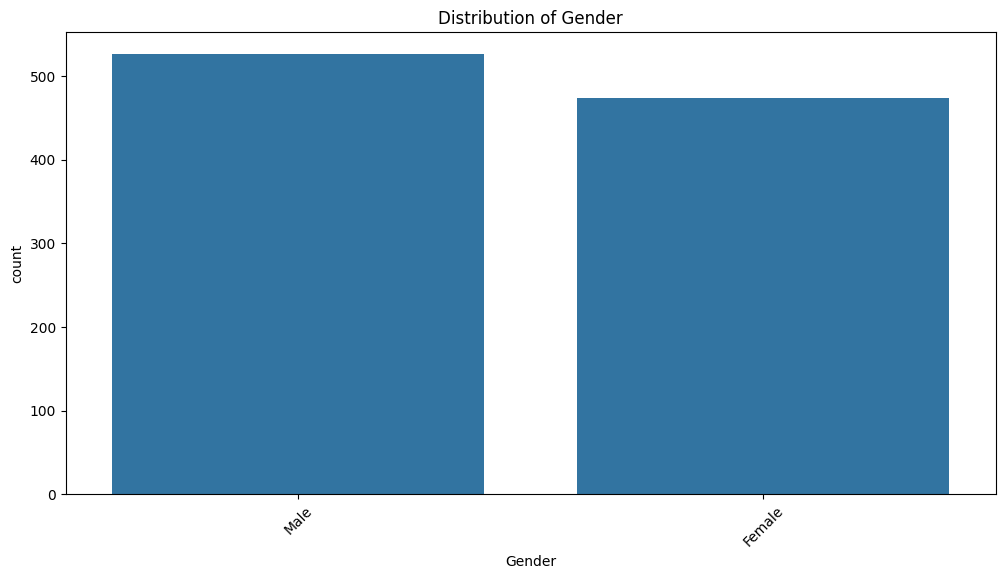
**plt.show()**

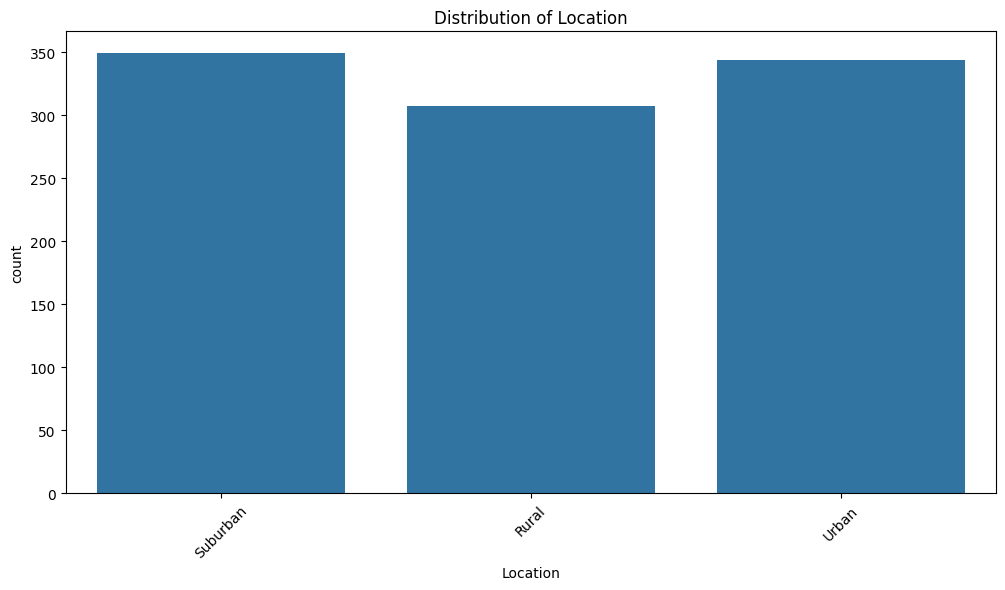
**sns.heatmap(df[num\_cols].corr(), annot=True, fmt='.2f', cmap='coolwarm')**

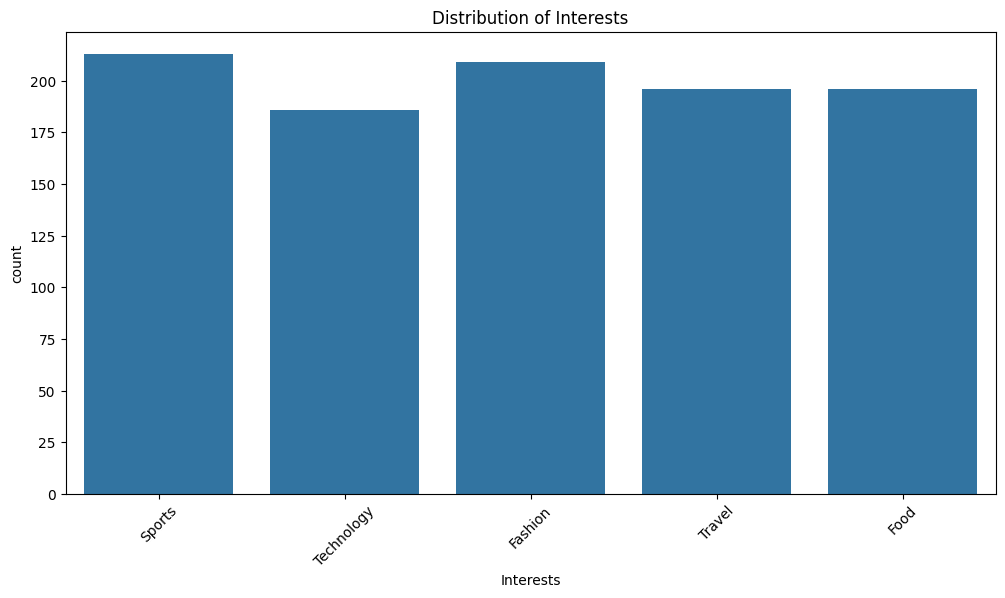
**plt.title('Correlation Heatmap')**

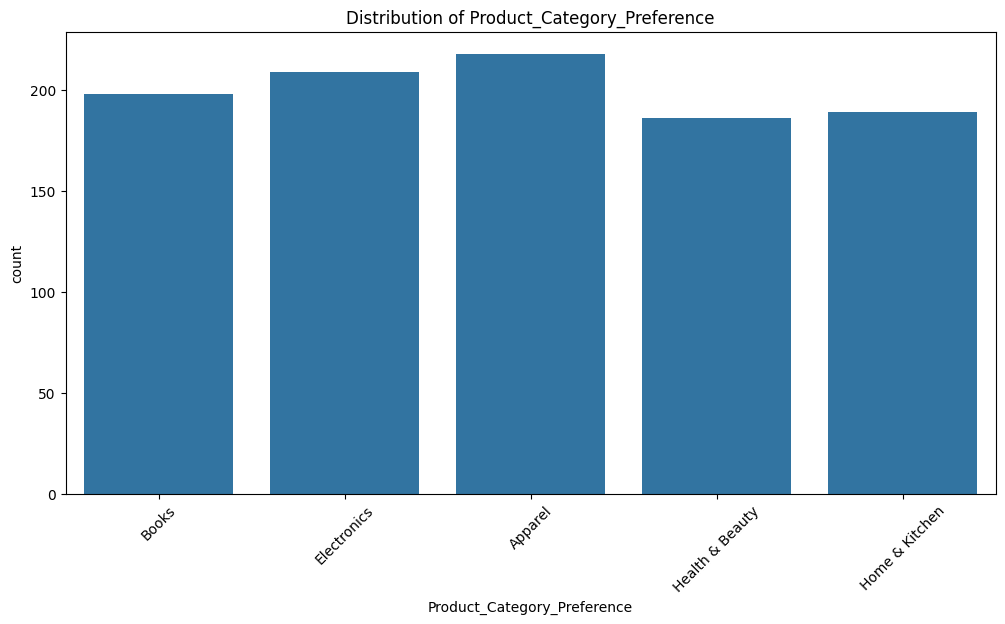
**plt.show()**

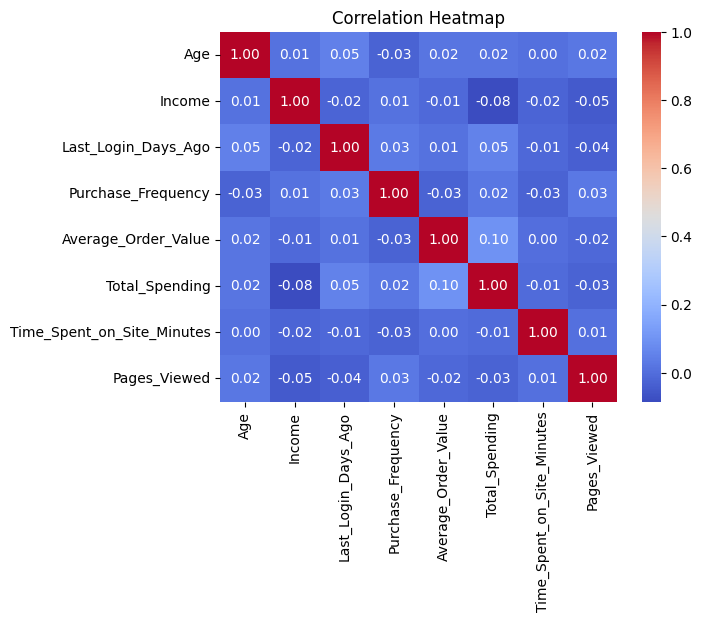
****

****

****

****

****

****

### 2.2.4. Mã hoá dữ liệu

Để huấn luyện mô hình, cần đảm bảo dữ liệu đã được mã hoá để máy tính có thể xử lí được. Cụ thể, các cột dữ liệu số sẽ được chuẩn hoá, để chúng xuất hiện trong khoảng (0, 1), tránh tạo ra trọng số cao hơn các nhóm còn lại. Các cột dữ liệu phân loại nhóm sẽ được sử dụng LabelEncoder() để mã hoá thành format máy tính có thể hiểu được.

Ngoài ra, nhóm chúng em sử dụng PCA để giảm số chiều dữ liệu, loại bỏ tín hiệu nhiễu và tăng tốc độ huấn luyện mô hình.

**from sklearn.preprocessing import StandardScaler, LabelEncoder**

**from sklearn.cluster import KMeans**

**from sklearn.decomposition import PCA**

**label\_encoder = {}**

**for col in cat\_cols:**

**le = LabelEncoder()**

**df[col] = le.fit\_transform(df[col])**

**label\_encoder[col] = le**

**scaler = StandardScaler()**

**df[num\_cols] = scaler.fit\_transform(df[num\_cols])**

**pca = PCA(n\_components=2)**

**pca\_components = pca.fit\_transform(df)**

# CHƯƠNG 3. XÂY DỰNG MÔ HÌNH

Để lựa chọn thuật toán phù hợp cho bài toán gợi ý sản phẩm người dùng, chương trình sẽ sử dụng 3 thuật toán bao gồm: KMeans, SVM, và Collaborative filtering - so sánh độ hiệu quả của các thuật toán trên và tạo ra một mô hình hybrid cho từng trường hợp khác nhau

## 3.1. Lựa chọn thuật toán

Vấn đề mà đề tài này đang giải quyết thuộc vào lớp bài toán hệ thống gợi ý (recommendation system) trong lĩnh vực Data Mining and Machine Learning (DMML). Đây là một trong những bài toán phổ biến và thực tiễn, nhằm mục tiêu cá nhân hóa trải nghiệm người dùng thông qua việc dự đoán và đề xuất những sản phẩm phù hợp với sở thích hoặc hành vi tiêu dùng trước đó.

Đã có nhiều thuật toán được đề xuất để giải quyết lớp bài toán này, từ các phương pháp thống kê truyền thống đến các mô hình học máy hiện đại. Trên cơ sở tham khảo từ các nghiên cứu, công trình liên quan như:

* Ricci et al., Recommender Systems Handbook (2015),
* [y] Aggarwal, C.C., Recommender Systems: The Textbook (2016),
* [z] Resnick & Varian, Recommender Systems (1997),
* cùng các giải pháp đã được công bố trên nền tảng thực nghiệm như Kaggle và GitHub,

Chúng em lựa chọn sử dụng các thuật toán sau đây để xây dựng hệ thống gợi ý cá nhân hóa:

* **KMeans:** Đây là một thuật toán phân cụm học không giám sát (unsupervised learning), hoạt động bằng cách chia tập dữ liệu thành k cụm khác nhau dựa trên độ tương đồng giữa các điểm (thường dùng khoảng cách Euclidean). Mỗi điểm dữ liệu sẽ được gán vào cụm có tâm gần nó nhất. Trong bối cảnh bài toán gợi ý, KMeans có thể được sử dụng để phân nhóm người dùng có hành vi hoặc sở thích tương đồng, từ đó đưa ra gợi ý sản phẩm phù hợp với từng nhóm cụ thể. Việc phân cụm giúp cá nhân hóa gợi ý mà không cần nhãn trước đó, đồng thời hỗ trợ hiểu sâu hơn về các phân khúc khách hàng tiềm năng.
* **Random Forest:** Là một mô hình học máy mạnh mẽ thuộc nhóm học có giám sát (supervised learning), Random Forest sử dụng tập hợp các cây quyết định (decision trees) để dự đoán hành vi hoặc phân loại người dùng. Thuật toán này giúp khai thác mối quan hệ phức tạp giữa các đặc trưng người dùng và sản phẩm, từ đó cải thiện độ chính xác của hệ thống gợi ý.
* **Collaborative Filtering:** Là kỹ thuật phổ biến trong hệ thống gợi ý, hoạt động dựa trên hành vi của người dùng thay vì nội dung sản phẩm. Collaborative Filtering có thể được áp dụng theo hướng User-based hoặc Item-based để dự đoán sở thích người dùng dựa trên cộng đồng người dùng tương tự hoặc các sản phẩm tương đồng. Trong đề tài này, chúng em triển khai theo hướng User-based

Sự kết hợp giữa các phương pháp trên giúp tăng độ linh hoạt và độ chính xác trong việc xây dựng hệ thống gợi ý, đồng thời cho phép đánh giá hiệu quả của từng thuật toán trong từng kịch bản cụ thể.

## 3.2. Các chỉ số đánh giá

**M1 – Accuracy (Độ chính xác tổng thể)**

* Đo lường tỷ lệ dự đoán đúng trên tổng số dự đoán.
* Phù hợp với các mô hình phân loại như SVM, Random Forest.
* Tuy nhiên, trong hệ thống gợi ý nơi dữ liệu thường mất cân bằng (imbalanced), độ chính xác có thể gây hiểu lầm nếu không đi kèm các chỉ số khác.
* Accuracy = Tổng số dự đoán đúng / Tổng số dự đoán

**M2 – F1-score**

* Là trung bình điều hòa giữa Precision và Recall.
* Cân bằng giữa độ chính xác đề xuất (precision) và độ bao phủ đề xuất (recall), phù hợp cho các bài toán mà cả hai yếu tố đều quan trọng.
* Đặc biệt hữu ích khi dữ liệu phân bố không đều giữa các lớp.
* F1=2 x (Precision x Recall) / Precision + Recall

​

**M3 – Precision (Độ chính xác đề xuất)**

* Cho biết trong số các sản phẩm được hệ thống gợi ý, bao nhiêu phần trăm là đúng với sở thích thực của người dùng.
* Precision cao có nghĩa là các đề xuất rất “gọn”, ít bị sai.
* Precision = True Positive / (True Positive + False Negative)

​

**M4 – Recall (Độ bao phủ đề xuất)**

* Đo lường khả năng hệ thống phát hiện được tất cả sản phẩm mà người dùng thực sự yêu thích.
* Recall cao có nghĩa là hệ thống đề xuất không bỏ sót các mặt hàng quan trọng.
* Recall = True Positive / (True Positive + False Positive)

**Chỉ số đánh giá mô hình KMeans:**

Do KMeans là một thuật toán không giám sát (unsupervised), không có nhãn thực tế để so sánh đúng/sai, nên hiệu quả mô hình được đánh giá bằng các chỉ số nội tại (intrinsic metrics) — phản ánh mức độ phân tách và chặt chẽ của các cụm.

Silhouette Score

* Ý nghĩa: Đo độ tương đồng của một điểm dữ liệu với cụm mà nó thuộc về so với cụm gần nhất khác.
* Giá trị: Từ -1 đến 1.
* ≈ 1: Cụm tách biệt tốt, phân cụm chất lượng cao.
* ≈ 0: Các cụm chồng chéo.
* < 0: Điểm có thể bị phân vào cụm sai.
* Ứng dụng: Dùng để đánh giá số cụm tối ưu, hoặc so sánh giữa các mô hình phân cụm khác nhau.

Davies–Bouldin Index (DBI)

* Ý nghĩa: Đo mức độ giống nhau giữa các cụm (sự chồng lấn).
* Giá trị: ≥ 0.
* Càng thấp → Cụm càng phân biệt rõ và mô hình càng hiệu quả.
* Ứng dụng: Dễ tính toán, phù hợp cho việc chọn mô hình có số cụm tốt nhất.

**Chỉ số đánh giá mô hình collaborative filtering**

Khi CF được sử dụng để dự đoán mức độ yêu thích, điểm số hoặc hành vi chi tiêu, mô hình cần được đánh giá bằng các chỉ số sai số hồi quy – thể hiện khoảng cách giữa giá trị dự đoán và thực tế.

**RMSE (Root Mean Squared Error)**

* Ý nghĩa: Sai số bình phương trung bình, rồi lấy căn bậc hai.
* Giá trị: ≥ 0. Càng nhỏ → mô hình càng chính xác.
* Đặc điểm: Nhạy với các giá trị ngoại lệ (outlier).

**MAE (Mean Absolute Error)**

* Ý nghĩa: Sai số tuyệt đối trung bình giữa giá trị dự đoán và thực tế.
* Giá trị: ≥ 0. Càng nhỏ → mô hình càng chính xác.
* Đặc điểm: Dễ hiểu, ít bị ảnh hưởng bởi ngoại lệ hơn RMSE.

## 3.3. Lựa chọn công nghệ:

Đề tài này, chúng em được giao sử dụng ngôn ngữ lập trình Python và các thư viện học máy liên quan như một ràng buộc kỹ thuật bắt buộc của đề tài. Python là một trong những ngôn ngữ phổ biến nhất hiện nay trong lĩnh vực khai phá dữ liệu (Data Mining) và học máy (Machine Learning) nhờ vào cú pháp đơn giản, thư viện đa dạng và cộng đồng phát triển mạnh.

Bên cạnh đó, công cụ chính được sử dụng là Jupyter Notebook, một môi trường hỗ trợ lập trình tương tác, trực quan hóa dữ liệu và trình bày kết quả một cách sinh động, thuận tiện cho việc mô phỏng hệ thống gợi ý.

Công nghệ Python hỗ trợ triển khai cho đề tài này bằng các thành phần sau:

* **Thành phần 1:** scikit-learn (sklearn): scikit-learn là một thư viện học máy mạnh mẽ trong Python, cung cấp các thuật toán phổ biến như: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), và Random Forest, cùng với các công cụ hỗ trợ chia tập dữ liệu, huấn luyện mô hình, đánh giá hiệu năng, chọn siêu tham số, v.v.
* Trong đề tài, scikit-learn được sử dụng để:
* Huấn luyện và đánh giá các mô hình học máy.
* So sánh độ chính xác giữa các thuật toán.
* Tạo tập huấn luyện và tập kiểm tra.
* Mã hóa dữ liệu của người dùng
* **Thành phần 2:** pandas và numpy: pandas và numpy là hai thư viện nền tảng hỗ trợ xử lý dữ liệu: pandas hỗ trợ thao tác với bảng dữ liệu dạng DataFrame, dễ dàng lọc, nhóm, thống kê và chuẩn hóa dữ liệu. numpy cung cấp các phép toán đại số tuyến tính và xử lý mảng hiệu năng cao.
* Trong đề tài, chúng em sử dụng các thư viện này để:
* Tiền xử lý dữ liệu người dùng.
* Thống kê và trực quan hóa dữ liệu ban đầu.
* Ngoài ra, các thư viện bổ trợ như matplotlib, seaborn và plotly cũng được sử dụng nhằm trực quan hóa dữ liệu và kết quả đánh giá mô hình, giúp minh họa rõ hơn hiệu quả của từng thuật toán trong hệ thống gợi ý.

## 3.4. Triển khai xây dựng mô hình

### 3.4.1. Áp dụng mô hình KMeans:

**Bước 1:** Xác định điểm K tối ưu sử dụng phương pháp Elbow, điểm K tối ưu sẽ nằm tại vị trí biểu đồ WCSS giảm độ dốc

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

label\_encoder = {}

for col in cat\_cols:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

label\_encoder[col] = le

scaler = StandardScaler()

df[num\_cols] = scaler.fit\_transform(df[num\_cols])

pca = PCA(n\_components=2)

pca\_components = pca.fit\_transform(df)

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, random\_state=42)

kmeans.fit(df)

wcss.append(kmeans.inertia\_)

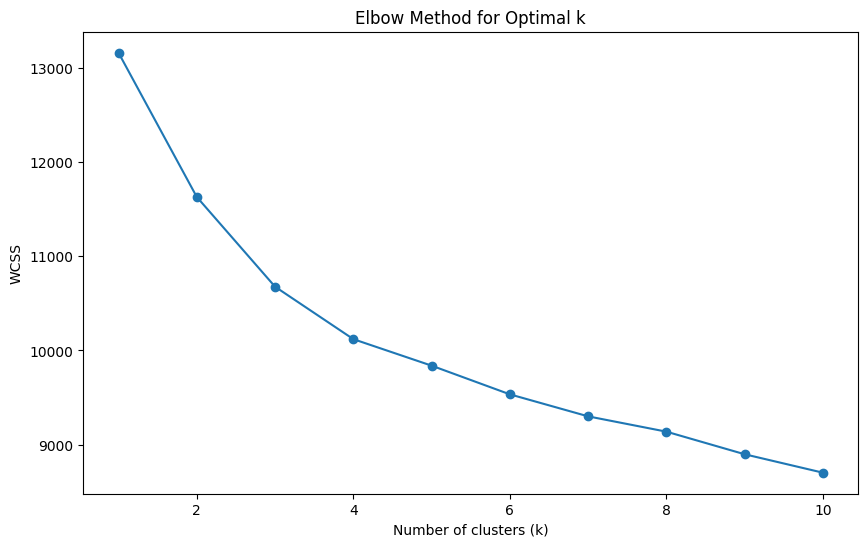
plt.figure(figsize=(10, 6))

plt.plot(range(1, 11), wcss, marker='o')

plt.title('Elbow Method for Optimal k')

plt.xlabel('Number of clusters (k)')

plt.ylabel('WCSS')



**Bước 2: Huấn luyện model sử dụng hệ số K**

optimal\_clusters = 4

kmeans = KMeans(n\_clusters=optimal\_clusters, random\_state=42)

df['Cluster'] = kmeans.fit\_predict(df)

plt.figure(figsize=(10, 10))

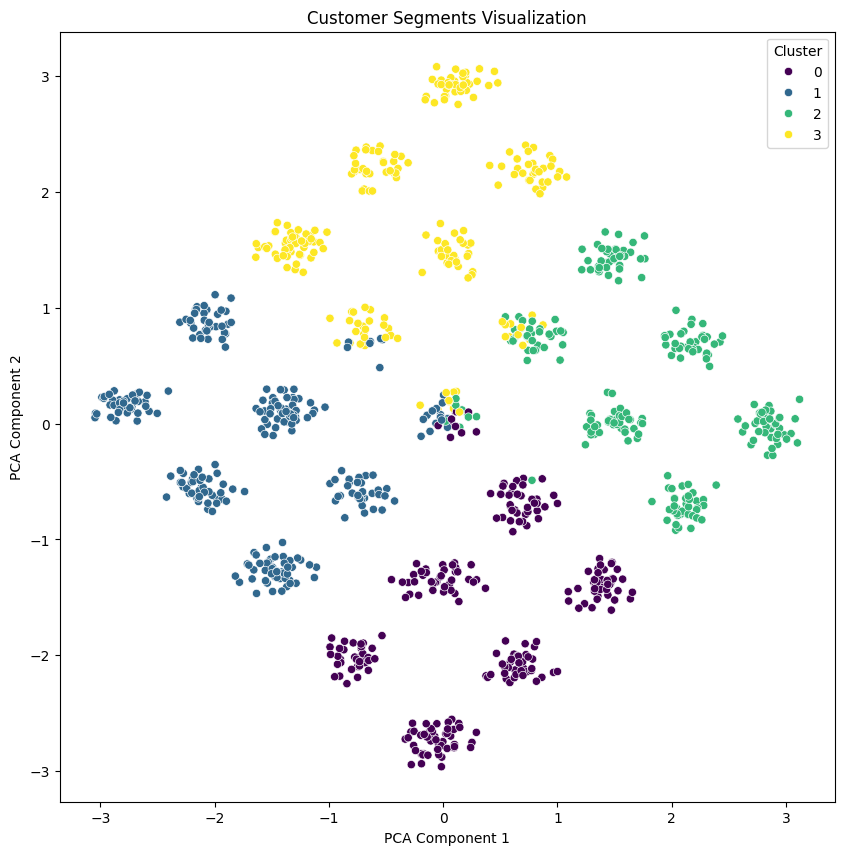
sns.scatterplot(x=pca\_components[:, 0], y=pca\_components[:, 1], hue=df['Cluster'], palette='viridis')

plt.title('Customer Segments Visualization')

plt.xlabel('PCA Component 1')

plt.ylabel('PCA Component 2')

plt.show()



### 3.4.2. Triển khai mô hình Collaborative filtering

**Bước 1: Phân cụm người bằng Kmeans**

Nhằm khám phá phân nhóm người dùng tiềm năng theo đặc điểm hành vi, thuật toán **KMeans** được áp dụng để phân cụm thành 4 nhóm.

kmeans = KMeans(n\_clusters=4, random\_state=42)

dataset['Cluster'] = kmeans.fit\_predict(dataset)

**Bước 2: Xây dựng hệ thống gợi ý**

* Dự đoán dựa trên tổng chi tiêu: Biến Total\_Spending được log-transform thành Log\_Total\_Spending để giảm ảnh hưởng của các giá trị ngoại lệ. Sau đó, sử dụng biến này làm thước đo đánh giá mức độ yêu thích của người dùng đối với từng loại sản phẩm.

import numpy as np

dataset1['Log\_Total\_Spending'] = np.log1p(dataset1['Total\_Spending'])

reader = Reader(rating\_scale=(dataset1['Log\_Total\_Spending'].min(), dataset1['Log\_Total\_Spending'].max()))

data = Dataset.load\_from\_df(dataset1[['User\_ID', 'Product\_Category\_Preference', 'Log\_Total\_Spending']], reader)

trainset, testset = train\_test\_split(data, test\_size=0.3, random\_state=42)

sim\_options = {

'name': 'cosine',

'user\_based': True

}

model = KNNBasic(sim\_options=sim\_options)

model.fit(trainset)

predictions = model.test(testset)

print(accuracy.rmse(predictions))

* Dự đoán dựa trên tần suất mua hàng: Ở biến thể thứ hai, chỉ giữ lại các giao dịch có Purchase\_Frequency > 0 và sử dụng tần suất này như thang điểm gợi ý.

from surprise import Dataset, Reader, KNNBasic

from surprise.model\_selection import train\_test\_split

from surprise import accuracy

# Use Purchase\_Frequency as the rating

filtered = dataset1[dataset1['Purchase\_Frequency'] > 0]

reader = Reader(rating\_scale=(filtered['Purchase\_Frequency'].min(), filtered['Purchase\_Frequency'].max()))

data = Dataset.load\_from\_df(filtered[['User\_ID', 'Product\_Category\_Preference', 'Purchase\_Frequency']], reader)

trainset, testset = train\_test\_split(data, test\_size=0.3, random\_state=42)

sim\_options = {

'name': 'cosine',

'user\_based': True

}

model = KNNBasic(sim\_options=sim\_options)

model.fit(trainset)

predictions = model.test(testset)

print(accuracy.rmse(predictions))

* Kết hợp cả tổng chi tiêu và tần suất mua hàng: Để tận dụng đồng thời cả **mức chi tiêu** và **tần suất mua hàng**, hai biến này được chuẩn hóa bằng MinMaxScaler, sau đó trung bình để tạo ra biến Combined\_Score.

from sklearn.preprocessing import MinMaxScaler

# Normalize features to [0, 1]

scaler = MinMaxScaler()

filtered['Norm\_Spending'] = scaler.fit\_transform(filtered[['Total\_Spending']])

filtered['Norm\_Frequency'] = scaler.fit\_transform(filtered[['Purchase\_Frequency']])

# Combine them (e.g., average)

filtered['Combined\_Score'] = (filtered['Norm\_Spending'] + filtered['Norm\_Frequency']) / 2

reader = Reader(rating\_scale=(filtered['Combined\_Score'].min(), filtered['Combined\_Score'].max()))

data = Dataset.load\_from\_df(filtered[['User\_ID', 'Product\_Category\_Preference', 'Combined\_Score']], reader)

trainset, testset = train\_test\_split(data, test\_size=0.3, random\_state=42)

sim\_options = {

'name': 'cosine',

'user\_based': True

}

model = KNNBasic(sim\_options=sim\_options)

model.fit(trainset)

predictions = model.test(testset)

print(accuracy.rmse(predictions))

### 3.4.3. Triển khai mô hình Random forest

**Bước 1: Chuẩn bị dữ liệu đầu vào**

Tập dữ liệu bao gồm các biến số (num\_cols) và biến phân loại (cat\_cols).

Cột đích Product\_Category\_Preference được tách ra làm nhãn (y) và loại bỏ khỏi tập đặc trưng (X) nếu nó tồn tại trong đó.

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

X = df[num\_cols + cat\_cols]

y = df['Product\_Category\_Preference']

if 'Product\_Category\_Preference' in X.columns:

X = X.drop(columns=['Product\_Category\_Preference'])

if 'Product\_Category\_Preference' in cat\_cols:

cat\_cols.remove('Product\_Category\_Preference')

**Bước 2: Xử lý dữ liệu phân loại**

Các cột phân loại (có kiểu object) được mã hóa số bằng LabelEncoder đã huấn luyện từ trước cho từng cột.

for col in cat\_cols:

if X[col].dtype == 'object':

X[col] = label\_encoder[col].transform(X[col])

**Bước 3: Chia dữ liệu và huấn luyện mô hình**

Dữ liệu được chia theo tỷ lệ 80% train – 20% test bằng train\_test\_split.

Sử dụng mô hình **Random Forest** với n\_estimators = 45 (tức 45 cây quyết định) và random\_state = 42 để đảm bảo tính tái lập.

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**rf = RandomForestClassifier(n\_estimators=45, random\_state=42)**

**rf.fit(X\_train, y\_train)**

**y\_pred = rf.predict(X\_test)**

**print(classification\_report(y\_test, y\_pred))**

## 3.5. Đánh giá chất lượng

### 3.5.1. Đánh giá chất lượng mô hình KMeans

| **Chỉ số đánh giá** | **Giá trị** |
| --- | --- |
| **Silhouette Score** | 0.5686 |
| **Davies-Bouldin Index** | 0.5018 |

**Silhouette Score = 0.5686** Cho thấy các điểm dữ liệu được phân cụm khá rõ ràng. Giá trị này nằm trong khoảng **trung bình-khá**, cho thấy các cụm có sự tách biệt nhất định, nhưng vẫn có khả năng chồng chéo giữa các cụm nhỏ.

**Davies-Bouldin Index = 0.5018.** Đây là một kết quả **rất tích cực**, vì **chỉ số càng thấp thì các cụm càng tách biệt và rõ ràng**. Mức 0.5 là tương đối tốt trong các bài toán phân cụm thực tế.

### 3.5.2. Đánh giá chất lượng mô hình Random Forest

| **Chỉ số** | **Giá trị** |
| --- | --- |
| **Accuracy** | 0.24 |
| **Precision (macro avg)** | 0.25 |
| **Recall (macro avg)** | 0.24 |
| **F1-score (macro avg)** | 0.24 |

| **Loại sản phẩm** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| **Apparel** | 0.21 | 0.33 | 0.26 | 39 |
| **Books** | 0.27 | 0.29 | 0.28 | 34 |
| **Electronics** | 0.26 | 0.27 | 0.27 | 44 |
| **Health & Beauty** | 0.19 | 0.18 | 0.19 | 38 |
| **Home & Kitchen** | 0.33 | 0.13 | 0.19 | 45 |

Độ chính xác tổng thể (accuracy) chỉ đạt 24%, cho thấy mô hình đang gặp khó khăn trong việc phân loại chính xác sở thích người dùng đối với các loại sản phẩm.

Precision, Recall và F1-score đều dao động quanh 0.24–0.26, cho thấy:

* Mô hình không phân biệt rõ được giữa các lớp sản phẩm.
* Có thể do phân bố lớp không đồng đều (class imbalance) hoặc đặc trưng đầu vào không đủ phân biệt.

Một số lớp như “Home & Kitchen” có precision = 0.33 nhưng recall lại rất thấp (0.13), tức là mô hình hay dự đoán nhầm thành lớp này nhưng lại bỏ sót nhiều trường hợp thật sự thuộc lớp đó.

### 3.5.3. Đánh giá chất lượng mô hình collaborative filtering

Mô hình User-based Collaborative Filtering được áp dụng nhằm dự đoán mức độ yêu thích hoặc tiềm năng mua sắm của người dùng đối với sản phẩm, dựa trên sự tương đồng giữa các người dùng.  
Dữ liệu đầu vào sử dụng các đặc trưng liên quan đến hành vi mua hàng, cụ thể:

* Total\_Spending – Tổng chi tiêu của người dùng.
* Purchase\_Frequency – Tần suất mua hàng.
* Tổng hợp cả hai đặc trưng ở trên.

|  | **Sử dụng total\_spending** | **Sử dụng purchase\_frequency** | **Kết hợp cả 2** |
| --- | --- | --- | --- |
| **RMSE** | 0.8324 | 2.6385 | 0.2289 |
| **MAE** | 0.6521 | 2.2609 | 0.1897 |

**Khi sử dụng riêng lẻ:**

* Total\_Spending cho kết quả tốt hơn so với Purchase\_Frequency, cả về RMSE lẫn MAE.
* Purchase\_Frequency tạo ra sai số lớn nhất, cho thấy đây có thể không phải là yếu tố đáng tin cậy nếu dùng riêng lẻ để suy đoán mức độ yêu thích.

**Khi kết hợp cả hai đặc trưng, mô hình cho kết quả vượt trội nhất, với:** RMSE = 0.2289 và MAE = 0.1897, cho thấy mức sai số rất thấp. Điều này chứng tỏ rằng việc kết hợp thông tin về cả chi tiêu và tần suất mua giúp mô hình hiểu sâu hơn về hành vi người dùng.

# CHƯƠNG 4. TÍCH HỢP MÔ HÌNH VÀO HỆ THỐNG

Mô hình cần được tích hợp vào chức năng cụ thể, thỏa mãn yêu cầu của đề bài

Viết theo hướng trình bày (1) sẽ tích hợp vào chức năng nào, mô tả chức năng; (2) tích hợp như thế nào; (3) kết quả tích hợp

## 4.1. Chức năng tích hợp: gợi ý sản phẩm người dùng

Mô hình hybrid được thiết kế để tích hợp vào chức năng gợi ý sản phẩm cá nhân hóa trên hệ thống thương mại điện tử.  
Trong đó:

* Với người dùng hiện tại (đã có lịch sử mua hàng hoặc tương tác): dùng Collaborative Filtering (CF).
* Với người dùng mới (new users) chưa có hành vi: dùng KMeans clustering để gán họ vào nhóm khách hàng có hành vi tương tự, và gợi ý sản phẩm phổ biến trong cụm đó.

Điều này giúp hệ thống vượt qua vấn đề cold-start, mở rộng khả năng gợi ý cho cả người dùng mới.

## 4.2. Phương thức tích hợp

Đối với người dùng mới:

* Thu thập thông tin hồ sơ: tuổi, giới tính, thu nhập, mức độ trung thành, khu vực, sở thích sản phẩm.
* Dùng mô hình KMeans đã huấn luyện từ trước để xác định cụm (cluster) mà người dùng mới phù hợp nhất.
* Lấy danh sách sản phẩm phổ biến nhất trong cụm đó để gợi ý.

Đối với người dùng hiện tại:

* Sử dụng User-based Collaborative Filtering để tìm người dùng tương tự (dựa trên hành vi).
* Dự đoán sản phẩm phù hợp nhất từ lịch sử người dùng tương tự.
* Gợi ý theo top-N sản phẩm có điểm dự đoán cao nhất.

import numpy as np

from sklearn.cluster import KMeans

from surprise import Dataset, KNNBasic

from surprise.model\_selection import train\_test\_split

from collections import defaultdict

def run\_user\_based\_cf(user\_id, trainset, cf\_model):

predictions = []

all\_items = trainset.all\_items()

item\_inner\_ids = [iid for iid in all\_items if not trainset.knows\_item(iid)]

for iid in item\_inner\_ids:

pred = cf\_model.predict(user\_id, trainset.to\_raw\_iid(iid))

predictions.append((iid, pred.est))

top\_n = sorted(predictions, key=lambda x: x[1], reverse=True)[:5]

return [trainset.to\_raw\_iid(iid) for iid, \_ in top\_n]

# === (2) GỢI Ý CHO NGƯỜI DÙNG MỚI BẰNG KMEANS ===

def run\_kmeans\_suggestion(user\_profile\_vector, kmeans\_model, cluster\_top\_products):

cluster\_id = kmeans\_model.predict([user\_profile\_vector])[0]

return cluster\_top\_products.get(cluster\_id, [])

def hybrid\_recommend(user\_id, user\_profile\_vector, trainset, cf\_model, kmeans\_model, cluster\_top\_products):

if trainset.knows\_user(user\_id):

return run\_user\_based\_cf(user\_id, trainset, cf\_model)

else:

return run\_kmeans\_suggestion(user\_profile\_vector, kmeans\_model, cluster\_top\_products)

## 4.3. Kết quả tích hợp

| **Người dùng** | **Mô hình áp dụng** | **Ưu điểm** |
| --- | --- | --- |
| Người dùng mới | **KMeans clustering** | Không cần lịch sử tương tác |
| Người dùng hiện tại | **User-based CF** | Gợi ý chính xác dựa trên hành vi |

Việc tích hợp hybrid giúp hệ thống:

* Linh hoạt hơn khi xử lý nhiều loại người dùng khác nhau
* Tăng độ phủ gợi ý, không bỏ sót người dùng mới
* Giảm rủi ro cold-start — vấn đề thường gặp ở hệ thống gợi ý truyền thống.

# KẾT LUẬN

Trong khuôn khổ đề tài, nhóm đã nghiên cứu, thiết kế và triển khai một hệ thống gợi ý sản phẩm dựa trên dữ liệu người dùng, kết hợp các phương pháp học máy truyền thống và hiện đại nhằm nâng cao hiệu quả gợi ý, đồng thời giải quyết bài toán cold-start – một vấn đề phổ biến trong các hệ thống gợi ý hiện nay. Kết quả đạt được được đối chiếu với các mục tiêu nghiên cứu ban đầu như sau:

## 1. Phân tích và tiền xử lý dữ liệu người dùng

Dữ liệu đầu vào bao gồm thông tin hồ sơ người dùng và hành vi mua sắm đã được tiến hành xử lý và biến đổi phù hợp với yêu cầu huấn luyện mô hình. Các bước như mã hóa biến phân loại, chuẩn hóa giá trị số, xử lý giá trị thiếu, và lựa chọn đặc trưng đã được thực hiện một cách hệ thống. Quá trình này giúp đảm bảo chất lượng dữ liệu và tăng độ chính xác cho các mô hình học máy được triển khai sau đó.

## 2. Xây dựng và đánh giá các mô hình học máy

Ba mô hình chính đã được áp dụng và đánh giá:

* **Random Forest** được sử dụng cho bài toán phân loại người dùng theo sở thích mặt hàng. Kết quả thử nghiệm trên tập dữ liệu kiểm thử cho thấy độ chính xác ở mức trung bình (**accuracy ~24%**), với chỉ số F1-score dao động từ 0.18 đến 0.28 giữa các lớp sản phẩm, phản ánh tính phức tạp và mất cân bằng trong dữ liệu phân loại.
* **KMeans Clustering** được triển khai để phân nhóm người dùng dựa trên các đặc trưng hồ sơ như độ tuổi, giới tính, thu nhập, mức độ trung thành,... nhằm phục vụ cho bài toán gợi ý đối với người dùng mới. Chất lượng phân cụm được đánh giá bằng hai chỉ số nội tại: **Silhouette Score = 0.5686** (mức khá tốt) và **Davies–Bouldin Index = 0.5018** (cho thấy cụm tách biệt tương đối rõ ràng).
* **User-based Collaborative Filtering** được sử dụng để cá nhân hóa đề xuất cho người dùng hiện tại dựa trên hành vi tương tác. Khi đánh giá mô hình theo hai chỉ số RMSE và MAE trên ba biến đầu vào khác nhau, phương án kết hợp giữa tổng chi tiêu và tần suất mua hàng cho kết quả tốt nhất (**RMSE = 0.2289**, **MAE = 0.1897**), phản ánh năng lực dự đoán đáng tin cậy.

## 3. Đề xuất mô hình hybrid kết hợp KMeans và Collaborative Filtering

Để khắc phục hạn chế của từng phương pháp riêng lẻ, đặc biệt là bài toán cold-start, đề tài đã xây dựng mô hình lai kết hợp giữa **KMeans** và **Collaborative Filtering**. Cụ thể, KMeans được sử dụng để xác định nhóm người dùng mới theo đặc trưng hồ sơ và gợi ý các sản phẩm phổ biến trong cụm tương ứng, trong khi CF được áp dụng cho người dùng hiện tại có sẵn dữ liệu hành vi để đưa ra gợi ý cá nhân hóa hơn. Mô hình này cho thấy tính linh hoạt cao, hiệu quả rõ rệt và dễ dàng tích hợp vào hệ thống thực tế.

## 4. Tích hợp mô hình vào chức năng gợi ý

Cuối cùng, đề tài đã xây dựng được quy trình tích hợp mô hình hybrid vào hệ thống gợi ý. Mô hình có thể tự động xác định loại người dùng (mới hoặc hiện tại) và lựa chọn phương pháp gợi ý phù hợp, từ đó tối ưu hóa trải nghiệm người dùng và tăng hiệu quả đề xuất sản phẩm. Cơ chế hoạt động được thiết kế theo hướng đơn giản, dễ triển khai và có khả năng mở rộng trong các hệ thống thương mại điện tử.

## Kết luận chung

Qua quá trình nghiên cứu và thử nghiệm, đề tài đã hoàn thành đầy đủ các mục tiêu đề ra. Hệ thống gợi ý được xây dựng có tính khả thi cao, có thể ứng dụng trong thực tiễn, đặc biệt là trong bối cảnh dữ liệu người dùng ngày càng phong phú và đa dạng. Bên cạnh đó, hướng tiếp cận kết hợp giữa các kỹ thuật học máy truyền thống và mô hình phân cụm mở ra nhiều tiềm năng phát triển, đặc biệt trong việc cá nhân hóa trải nghiệm người dùng và giải quyết các thách thức như cold-start và dữ liệu thưa thớt.

## TÀI LIỆU THAM KHẢO

Breese, J. S., Heckerman, D., & Kadie, C. (1998). *Empirical analysis of predictive algorithms for collaborative filtering*. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98), 43–52.

Çano, E., & Morisio, M. (2019). *Hybrid recommender systems: A systematic literature review*. arXiv preprint arXiv:1901.03888.<https://arxiv.org/abs/1901.03888>

Panteli, D., & Boutsinas, B. (2023). *A hybrid semantic recommender system based on an improved clustering approach*. Algorithms, 16(4), 182. https://www.mdpi.com/1999-4893/16/4/182

Zhang, Z., Chen, T., Li, X., & Yang, Y. (2021). *A hybrid recommendation model based on K-means and matrix factorization*. Journal of Intelligent & Fuzzy Systems, 40(5), 1–10. https://doi.org/10.3233/JIFS-201210

Lin, C., & Lin, J. (2019). *Collaborative filtering recommendation model based on K-means clustering*. International Journal of Computer Applications, 182(5), 10–15.

**PHỤ LỤC**