import pandas as pd  
  
# Load the shopping trends dataset  
df = pd.read\_csv('shopping\_trends.csv')  
  
# Check the structure of your data  
print(df.head())

Customer Age Gender Item Purchased Category Purchase Amount (USD) \  
0 Cust 1 55 Male Blouse Clothing 53   
1 Cust 2 19 Male Sweater Clothing 64   
2 Cust 3 50 Male Jeans Clothing 73   
3 Cust 4 21 Male Sandals Footwear 90   
4 Cust 5 45 Male Blouse Clothing 49   
  
 Location Size Color Season Rating Subscription Status \  
0 Kentucky L Gray Winter 3.1 Yes   
1 Maine L Maroon Winter 3.1 Yes   
2 Massachusetts S Maroon Spring 3.1 Yes   
3 Rhode Island M Maroon Spring 3.5 Yes   
4 Oregon M Turquoise Spring 2.7 Yes   
  
 Payment Method Shipping Type Discount Applied Promo Code Used \  
0 Credit Card Express Yes Yes   
1 Bank Transfer Express Yes Yes   
2 Cash Free Shipping Yes Yes   
3 PayPal Next Day Air Yes Yes   
4 Cash Free Shipping Yes Yes   
  
 Previous Purchases Preferred Payment Method Frequency of Purchases   
0 14 Venmo Fortnightly   
1 2 Cash Fortnightly   
2 23 Credit Card Weekly   
3 49 PayPal Weekly   
4 31 PayPal Annually

print(df.tail())

Customer Age Gender Item Purchased Category \  
3895 Cust 3896 40 Female Hoodie Clothing   
3896 Cust 3897 52 Female Backpack Accessories   
3897 Cust 3898 46 Female Belt Accessories   
3898 Cust 3899 44 Female Shoes Footwear   
3899 Cust 3900 52 Female Handbag Accessories   
  
 Purchase Amount (USD) Location Size Color Season Rating \  
3895 28 Virginia L Turquoise Summer 4.2   
3896 49 Iowa L White Spring 4.5   
3897 33 New Jersey L Green Spring 2.9   
3898 77 Minnesota S Brown Summer 3.8   
3899 81 California M Beige Spring 3.1   
  
 Subscription Status Payment Method Shipping Type Discount Applied \  
3895 No Cash 2-Day Shipping No   
3896 No PayPal Store Pickup No   
3897 No Credit Card Standard No   
3898 No PayPal Express No   
3899 No Bank Transfer Store Pickup No   
  
 Promo Code Used Previous Purchases Preferred Payment Method \  
3895 No 32 Venmo   
3896 No 41 Bank Transfer   
3897 No 24 Venmo   
3898 No 24 Venmo   
3899 No 33 Venmo   
  
 Frequency of Purchases   
3895 Weekly   
3896 Bi-Weekly   
3897 Quarterly   
3898 Weekly   
3899 Quarterly

print(df.info())

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3900 entries, 0 to 3899  
Data columns (total 19 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Customer 3900 non-null object   
 1 Age 3900 non-null int64   
 2 Gender 3900 non-null object   
 3 Item Purchased 3900 non-null object   
 4 Category 3900 non-null object   
 5 Purchase Amount (USD) 3900 non-null int64   
 6 Location 3900 non-null object   
 7 Size 3900 non-null object   
 8 Color 3900 non-null object   
 9 Season 3900 non-null object   
 10 Rating 3900 non-null float64  
 11 Subscription Status 3900 non-null object   
 12 Payment Method 3900 non-null object   
 13 Shipping Type 3900 non-null object   
 14 Discount Applied 3900 non-null object   
 15 Promo Code Used 3900 non-null object   
 16 Previous Purchases 3900 non-null int64   
 17 Preferred Payment Method 3900 non-null object   
 18 Frequency of Purchases 3900 non-null object   
dtypes: float64(1), int64(3), object(15)  
memory usage: 579.0+ KB  
None

print(df.describe())

Age Purchase Amount (USD) Rating Previous Purchases  
count 3900.000000 3900.000000 3900.000000 3900.000000  
mean 44.068462 59.764359 3.749949 25.351538  
std 15.207589 23.685392 0.716223 14.447125  
min 18.000000 20.000000 2.500000 1.000000  
25% 31.000000 39.000000 3.100000 13.000000  
50% 44.000000 60.000000 3.700000 25.000000  
75% 57.000000 81.000000 4.400000 38.000000  
max 70.000000 100.000000 5.000000 50.000000

print(df.shape)

(3900, 19)

user\_counts = df['Customer'].value\_counts()  
print(user\_counts)

Cust 1 1  
Cust 2621 1  
Cust 2593 1  
Cust 2594 1  
Cust 2595 1  
 ..  
Cust 1305 1  
Cust 1306 1  
Cust 1307 1  
Cust 1308 1  
Cust 3900 1  
Name: Customer, Length: 3900, dtype: int64

user\_item\_matrix = df.pivot(index='Customer', columns='Category', values='Rating').fillna(0)

from sklearn.metrics.pairwise import cosine\_similarity  
  
# Calculate user-user similarity using cosine similarity  
user\_similarity = cosine\_similarity(user\_item\_matrix)  
  
# Define a function to make recommendations for a user  
def recommend\_products(Customer, user\_item\_matrix, user\_similarity, num\_recommendations=5):  
 user\_idx = user\_item\_matrix.index.get\_loc(Customer)  
 user\_similarities = user\_similarity[user\_idx]  
 product\_scores = user\_item\_matrix.values.T.dot(user\_similarities)  
  
 # Sort products by score in descending order and get top recommendations  
 recommended\_product\_indices = product\_scores.argsort()[::-1][:num\_recommendations]  
 recommended\_products = user\_item\_matrix.columns[recommended\_product\_indices]  
  
 return recommended\_products  
  
# Example: Get product recommendations for a specific user  
Customer= 'Cust 1'  
recommended\_products = recommend\_products(Customer, user\_item\_matrix, user\_similarity)  
print(f"Recommended products for {Customer}:", recommended\_products)

Recommended products for Cust 1: Index(['Clothing', 'Outerwear', 'Footwear', 'Accessories'], dtype='object', name='Category')

import pandas as pd  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.metrics.pairwise import cosine\_similarity  
  
# Load the shopping trends dataset  
#df = pd.read\_csv('shopping\_trends.csv')  
  
# Create a TF-IDF vectorizer  
vectorizer = TfidfVectorizer(stop\_words='english')  
  
# Fit and transform the 'Category' column  
tfidf\_matrix = vectorizer.fit\_transform(df['Category'].values.astype('U'))  
  
def recommend\_product(input\_text):  
 # Transform the input text using the vectorizer  
 input\_vector = vectorizer.transform([input\_text])  
  
 # Calculate the cosine similarity between input and dataset  
 similarities = cosine\_similarity(input\_vector, tfidf\_matrix)  
  
 # Get the index of the most similar product  
 most\_similar\_index = similarities.argmax()  
  
 # Return the details of the most similar product  
 most\_similar\_product = df.iloc[most\_similar\_index]  
 return most\_similar\_product  
  
# Test the function  
user\_input = input("Enter your text: ")  
recommended\_product = recommend\_product(user\_input)  
print("Recommended Product:")  
print("Cateogory:", recommended\_product['Category'])  
print("User:", recommended\_product['Customer'])  
print("Rating:", recommended\_product['Rating'])

Enter your text: Blouse  
Recommended Product:  
Cateogory: Clothing  
User: Cust 1  
Rating: 3.1

from sklearn.model\_selection import train\_test\_split  
# Split the data into training and testing sets  
train\_data, test\_data = train\_test\_split(df, test\_size=0.2, random\_state=42)

from sklearn.neighbors import NearestNeighbors  
# Create a k-NN classifier with k=5 (you can adjust k as needed)  
# Create a user-item interaction matrix  
user\_item\_matrix = df.pivot(index='Customer', columns='Category', values='Rating').fillna(0)  
  
# Create a Nearest Neighbors model  
knn = NearestNeighbors(n\_neighbors=5, metric='cosine', algorithm='brute', n\_jobs=-1)  
knn.fit(user\_item\_matrix)

NearestNeighbors(algorithm='brute', metric='cosine', n\_jobs=-1)

from sklearn.metrics import mean\_squared\_error  
import numpy as np  
import warnings  
warnings.filterwarnings("ignore", category=UserWarning)  
  
# Predict ratings for test data  
user\_indices = user\_item\_matrix.index  
product\_indices = user\_item\_matrix.columns  
  
predicted\_ratings = []  
  
for \_, row in test\_data.iterrows():  
 user = row['Customer']  
 product = row['Category']  
 rating = row['Rating']  
  
 if user in user\_indices and product in product\_indices:  
 user\_index = user\_indices.get\_loc(user)  
 product\_index = product\_indices.get\_loc(product)  
  
 # Find k-nearest neighbors  
 distances, indices = knn.kneighbors([user\_item\_matrix.iloc[user\_index].values], 5)  
 neighbor\_ratings = user\_item\_matrix.iloc[indices[0]].values  
 predicted\_rating = np.mean(neighbor\_ratings[:, product\_index])  
  
 predicted\_ratings.append(predicted\_rating)  
  
# Calculate RMSE  
actual\_ratings = test\_data['Rating'].values  
rmse = np.sqrt(mean\_squared\_error(actual\_ratings, predicted\_ratings))  
print("Root Mean Squared Error (RMSE):", rmse)

Root Mean Squared Error (RMSE): 0.770154496655021

import matplotlib.pyplot as plt  
  
def visualize\_recommendations(Customer, recommended\_products, product\_scores):  
 plt.figure(figsize=(10, 6))  
 plt.barh(recommended\_products, product\_scores, color='skyblue')  
 plt.xlabel('Recommendation Score')  
 plt.title(f'Recommended Products for {Customer}')  
 plt.gca().invert\_yaxis() # Invert the y-axis to show the top recommendations at the top  
 plt.show()  
  
# Example: Get product recommendations for a specific user  
Customer = 'Cust 1'  
recommended\_products = recommend\_products(Customer, user\_item\_matrix, user\_similarity)  
product\_scores = user\_item\_matrix.loc[Customer].values  
  
visualize\_recommendations(Customer, recommended\_products, product\_scores)

