Piyush_Chouhan_Lookalike

January 24, 2025

1 Piyush_Chouhan_Lookalike.ipynb

2 Task 2: Lookalike Model

The goal of this notebook is to: 1. Use customer and transaction data to recommend 3 similar customers for each user. 2. Use a similarity measure to assign scores to recommended customers. 3. Generate a CSV file with recommendations for the first 20 customers.

---

[2]:
```
# Import necessary libraries
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler
import numpy as np
```

2.1 Step 2: Load the Data

We will load the `Customers.csv`, `Products.csv`, and `Transactions.csv` datasets.

[3]:
```
# Load datasets
customers = pd.read_csv('Customers.csv')
transactions = pd.read_csv('Transactions.csv')
products = pd.read_csv('Products.csv')

# Display the first few rows of each dataset
print("Customers:")
print(customers.head())

print("\nTransactions:")
print(transactions.head())

print("\nProducts:")
print(products.head())
```

```
Customers:
  CustomerID        CustomerName         Region  SignupDate
0      C0001    Lawrence Carroll  South America  2022-07-10
1      C0002      Elizabeth Lutz           Asia  2022-02-13
```

```
2      C0003      Michael Rivera  South America  2024-03-07
3      C0004  Kathleen Rodriguez  South America  2022-10-09
4      C0005         Laura Weber           Asia  2022-08-15

Transactions:
  TransactionID CustomerID ProductID      TransactionDate  Quantity  \
0        T00001      C0199      P067  2024-08-25 12:38:23         1
1        T00112      C0146      P067  2024-05-27 22:23:54         1
2        T00166      C0127      P067  2024-04-25 07:38:55         1
3        T00272      C0087      P067  2024-03-26 22:55:37         2
4        T00363      C0070      P067  2024-03-21 15:10:10         3

   TotalValue   Price
0      300.68  300.68
1      300.68  300.68
2      300.68  300.68
3      601.36  300.68
4      902.04  300.68

Products:
  ProductID              ProductName     Category   Price
0      P001     ActiveWear Biography        Books  169.30
1      P002    ActiveWear Smartwatch  Electronics  346.30
2      P003  ComfortLiving Biography        Books   44.12
3      P004            BookWorld Rug   Home Decor   95.69
4      P005           TechPro T-Shirt     Clothing  429.31
```

2.2 Step 3: Preprocess the Data

We will: 1. Merge the datasets to create a complete customer profile. 2. Aggregate the data to generate a transaction history per customer.

[4]:
```python
# Merge transactions with products
merged_data = pd.merge(transactions, products, on='ProductID')

# Aggregate transaction data to get total value per customer for each product␣
↪category
customer_profile = merged_data.groupby(['CustomerID',␣
↪'Category'])['TotalValue'].sum().unstack(fill_value=0)

# Display the aggregated customer profile
print("Customer Profile:")
print(customer_profile.head())
```

```
Customer Profile:
Category       Books  Clothing  Electronics  Home Decor
CustomerID
C0001         114.60      0.00      2827.30      412.62
```

```
C0002          0.00    1025.46        0.00      837.28
C0003          0.00     122.36     1385.20     1217.82
C0004       1888.48       0.00     1355.74     2110.66
C0005          0.00       0.00     1180.38      853.86
```

2.3 Step 4: Compute Similarities

We will: 1. Normalize the customer profiles using `StandardScaler`. 2. Use cosine similarity to compute pairwise similarity scores between customers.

[5]:
```python
# Normalize the customer profile data
scaler = StandardScaler()
normalized_data = scaler.fit_transform(customer_profile)

# Compute cosine similarity between customers
similarity_matrix = cosine_similarity(normalized_data)

# Create a DataFrame for the similarity matrix
similarity_df = pd.DataFrame(similarity_matrix, index=customer_profile.index,␣
 ↪columns=customer_profile.index)

# Display a sample of the similarity matrix
print("Similarity Matrix:")
print(similarity_df.head())
```

```
Similarity Matrix:
CustomerID     C0001     C0002     C0003     C0004     C0005     C0006  \
CustomerID
C0001       1.000000 -0.402215  0.648350  0.043313  0.661203 -0.960708
C0002      -0.402215  1.000000  0.175482 -0.446094  0.257825  0.235584
C0003       0.648350  0.175482  1.000000  0.328565  0.932178 -0.734670
C0004       0.043313 -0.446094  0.328565  1.000000  0.092857 -0.005891
C0005       0.661203  0.257825  0.932178  0.092857  1.000000 -0.814067

CustomerID     C0007     C0008     C0009     C0010  …     C0191     C0192  \
CustomerID                                          …
C0001       0.637812 -0.268011  0.171019 -0.381244  … -0.059019  0.830892
C0002       0.166689  0.470266  0.588281  0.703980  … -0.527737 -0.050379
C0003       0.996881  0.202597  0.198752 -0.372100  … -0.448319  0.462407
C0004       0.347577  0.112209 -0.725347 -0.913367  …  0.068221 -0.451726
C0005       0.945695 -0.065768  0.508871 -0.144782  … -0.240979  0.657526

CustomerID     C0193     C0194     C0195     C0196     C0197     C0198  \
CustomerID
C0001      -0.382490  0.576188 -0.030011 -0.719949  0.461473 -0.173126
C0002      -0.374197  0.036706  0.426021  0.311911  0.409603  0.762741
C0003      -0.605579  0.316061  0.610345 -0.146110  0.919496 -0.064460
C0004       0.104261 -0.379710  0.422922  0.433344  0.165504 -0.839495
```

```
C0005      -0.431373  0.220120  0.349060 -0.392847  0.967956  0.213940

CustomerID     C0199     C0200
CustomerID
C0001       0.165667 -0.756913
C0002       0.522735  0.116516
C0003       0.840167 -0.867465
C0004       0.330655 -0.335804
C0005       0.817298 -0.918261

[5 rows x 199 columns]
```

2.4 Step 5: Recommend Top 3 Lookalike Customers

We will: 1. Find the top 3 most similar customers for each customer. 2. Store the results in a dictionary.

[6]:
```python
# Function to get top 3 similar customers
def get_top_3_similar(customer_id):
    # Exclude the customer itself (similarity = 1.0)
    similar_customers = similarity_df[customer_id].drop(customer_id)
    # Get the top 3 similar customers and their scores
    top_3 = similar_customers.nlargest(3).reset_index()
    top_3.columns = ['CustomerID', 'SimilarityScore']
    return top_3

# Generate recommendations for the first 20 customers
recommendations = {}
for customer_id in similarity_df.index[:20]:
    recommendations[customer_id] = get_top_3_similar(customer_id).values.
 ↪tolist()

# Display recommendations for the first few customers
print("Lookalike Recommendations:")
for key, value in recommendations.items():
    print(f"{key}: {value}")
```

```
Lookalike Recommendations:
C0001: [['C0091', 0.9888478853919913], ['C0069', 0.9843439691570108], ['C0184',
0.9785619388073006]]
C0002: [['C0159', 0.9795105096869298], ['C0036', 0.9567623389803885], ['C0134',
0.9079308367148381]]
C0003: [['C0007', 0.9968810093511232], ['C0085', 0.9640463999694346], ['C0166',
0.9603809339360593]]
C0004: [['C0075', 0.9832140510191636], ['C0090', 0.9205815444205002], ['C0065',
0.8848698960012958]]
C0005: [['C0197', 0.9679556789053673], ['C0085', 0.9638213905900722], ['C0166',
0.9498424445847362]]
```

```
C0006: [['C0169', 0.9704006342294953], ['C0185', 0.9294489990803511], ['C0081',
0.9274392104956387]]
C0007: [['C0003', 0.9968810093511232], ['C0085', 0.9790544159375512], ['C0166',
0.9584810984308859]]
C0008: [['C0143', 0.9757288678444452], ['C0158', 0.953510746297948], ['C0170',
0.9505172118501191]]
C0009: [['C0032', 0.9803022627726677], ['C0058', 0.9683007773908333], ['C0150',
0.9614456236906473]]
C0010: [['C0029', 0.9976752433133421], ['C0062', 0.9804016075245915], ['C0111',
0.9801805189388902]]
C0011: [['C0117', 0.963973419326239], ['C0016', 0.9632077461792568], ['C0074',
0.9510195409647744]]
C0012: [['C0148', 0.9884792503556856], ['C0152', 0.9539841832634688], ['C0113',
0.9513290068238263]]
C0013: [['C0046', 0.9814947146937686], ['C0099', 0.9465853217128836], ['C0117',
0.919516728582664]]
C0014: [['C0151', 0.9979207390553534], ['C0097', 0.9927491228970107], ['C0060',
0.9880496411636847]]
C0015: [['C0071', 0.976486876158439], ['C0121', 0.9723159872874918], ['C0025',
0.9688259324918211]]
C0016: [['C0011', 0.9632077461792568], ['C0117', 0.9247869590037776], ['C0074',
0.8760745705780278]]
C0017: [['C0179', 0.8911139153655481], ['C0122', 0.8887860009610975], ['C0064',
0.8670294480943582]]
C0018: [['C0023', 0.9823466564399504], ['C0051', 0.9772356807245444], ['C0168',
0.9765757848964493]]
C0019: [['C0035', 0.9383055966012707], ['C0177', 0.9136577573845739], ['C0070',
0.8877945815992546]]
C0020: [['C0130', 0.9794407284463068], ['C0094', 0.955158720476758], ['C0032',
0.9273591770429713]]
```

2.5 Step 6: Save Recommendations to CSV

We will save the recommendations as `Piyush_Chouhan_Lookalike.csv`.

[7]:
```python
# Prepare data for CSV export
recommendation_list = []
for customer, similar_customers in recommendations.items():
    for similar_customer, score in similar_customers:
        recommendation_list.append([customer, similar_customer, score])

# Create a DataFrame
recommendation_df = pd.DataFrame(recommendation_list, columns=['CustomerID',␣
 ↪'SimilarCustomerID', 'SimilarityScore'])

# Save to CSV
recommendation_df.to_csv('Piyush_Chouhan_Lookalike.csv', index=False)
```

```python
print("Recommendations saved to Piyush_Chouhan_Lookalike.csv")
```

```
Recommendations saved to Piyush_Chouhan_Lookalike.csv
```