Dnyanesh_Shinde_Lookalike

January 27, 2025

[1]:
```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler
```

[2]:
```python
# Load datasets
customers = pd.read_csv("Customers.csv")
products = pd.read_csv("Products.csv")
transactions = pd.read_csv("Transactions.csv")
```

[3]:
```python
# Display basic information
print("Basic information in Customers.csv:\n",customers.info())
print("\nBasic information in Products.csv:\n",products.info())
print("\nBasic information in Transactions.csv:\n",transactions.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
 2   Region        200 non-null    object
 3   SignupDate    200 non-null    object
dtypes: object(4)
memory usage: 6.4+ KB
Basic information in Customers.csv:
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ProductID    100 non-null    object
 1   ProductName  100 non-null    object
```

```
 2   Category     100 non-null    object
 3   Price        100 non-null    float64
dtypes: float64(1), object(3)
memory usage: 3.3+ KB

Basic information in Products.csv:
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    1000 non-null   object
 1   CustomerID       1000 non-null   object
 2   ProductID        1000 non-null   object
 3   TransactionDate  1000 non-null   object
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.8+ KB

Basic information in Transactions.csv:
 None
```

[4]:
```
# Step 1: Merge Data for Feature Engineering
# Merge transactions with customers and products
merged_data = transactions.merge(customers, on="CustomerID").merge(products,␣
 ↪on="ProductID")
merged_data
```

[4]:
```
    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
..            …          …         …                    …        …
995        T00630      C0031      P093  2024-10-08 23:58:14         2
996        T00672      C0165      P044  2024-07-28 00:09:49         4
997        T00711      C0165      P044  2024-06-11 15:51:14         4
998        T00878      C0165      P044  2024-09-24 21:15:21         3
999        T00157      C0169      P044  2024-11-09 09:07:36         2

     TotalValue  Price_x      CustomerName         Region  SignupDate  \
0        300.68   300.68   Andrea Jenkins         Europe  2022-12-03
1        300.68   300.68  Brittany Harvey           Asia  2024-09-04
```

```
2        300.68   300.68  Kathryn Stevens         Europe  2024-04-04
3        601.36   300.68  Travis Campbell  South America  2024-04-11
4        902.04   300.68    Timothy Perez         Europe  2022-03-15
..          …        …             …              …         …
995      609.88   304.94      Tina Miller  South America  2024-04-11
996       75.28    18.82     Juan Mcdaniel  South America  2022-04-09
997       75.28    18.82     Juan Mcdaniel  South America  2022-04-09
998       56.46    18.82     Juan Mcdaniel  South America  2022-04-09
999       37.64    18.82     Jennifer Shaw  South America  2023-04-13

                         ProductName     Category  Price_y
0    ComfortLiving Bluetooth Speaker  Electronics   300.68
1    ComfortLiving Bluetooth Speaker  Electronics   300.68
2    ComfortLiving Bluetooth Speaker  Electronics   300.68
3    ComfortLiving Bluetooth Speaker  Electronics   300.68
4    ComfortLiving Bluetooth Speaker  Electronics   300.68
..                               …          …        …
995                     TechPro Vase   Home Decor   304.94
996          ActiveWear Running Shoes     Clothing    18.82
997          ActiveWear Running Shoes     Clothing    18.82
998          ActiveWear Running Shoes     Clothing    18.82
999          ActiveWear Running Shoes     Clothing    18.82

[1000 rows x 13 columns]
```

[5]:
```python
merged_data = merged_data.rename(columns={'Price_x': 'Price'})  # Use this if Price_x is relevant
```

[6]:
```python
# Aggregate transactional data for each customer
customer_features = merged_data.groupby('CustomerID').agg(
    total_spent=('TotalValue', 'sum'),
    total_quantity=('Quantity', 'sum'),
    avg_price_per_item=('Price', 'mean'),
    num_transactions=('TransactionID', 'count'),
).reset_index()
customer_features
```

[6]:
```
    CustomerID  total_spent  total_quantity  avg_price_per_item  \
0        C0001      3354.52              12          278.334000
1        C0002      1862.74              10          208.920000
2        C0003      2725.38              14          195.707500
3        C0004      5354.88              23          240.636250
4        C0005      2034.24               7          291.603333
..         …           …              …                  …
194      C0196      4982.88              12          416.992500
195      C0197      1928.65               9          227.056667
196      C0198       931.83               3          239.705000
```

```
197       C0199      1979.28                9          250.610000
198       C0200      4758.60               16          296.506000

     num_transactions
0                   5
1                   4
2                   4
3                   8
4                   3
..                ...
194                 4
195                 3
196                 2
197                 4
198                 5

[199 rows x 5 columns]
```

[7]:

```python
# Add customer demographic features
customer_features = customer_features.merge(customers[['CustomerID',␣
↪'Region']], on='CustomerID')
customer_features
```

[7]:

```
    CustomerID  total_spent  total_quantity  avg_price_per_item  \
0        C0001      3354.52              12          278.334000
1        C0002      1862.74              10          208.920000
2        C0003      2725.38              14          195.707500
3        C0004      5354.88              23          240.636250
4        C0005      2034.24               7          291.603333
..         ...          ...             ...                 ...
194      C0196      4982.88              12          416.992500
195      C0197      1928.65               9          227.056667
196      C0198       931.83               3          239.705000
197      C0199      1979.28               9          250.610000
198      C0200      4758.60              16          296.506000

     num_transactions         Region
0                   5  South America
1                   4           Asia
2                   4  South America
3                   8  South America
4                   3           Asia
..                ...            ...
194                 4         Europe
195                 3         Europe
196                 2         Europe
197                 4         Europe
```

```
198                 5          Asia

[199 rows x 6 columns]
```

[8]:
```python
# One-hot encode categorical features
customer_features_encoded = pd.get_dummies(customer_features,␣
↪columns=['Region'], drop_first=True)
customer_features_encoded
```

[8]:
```
    CustomerID  total_spent  total_quantity  avg_price_per_item  \
0        C0001      3354.52              12          278.334000
1        C0002      1862.74              10          208.920000
2        C0003      2725.38              14          195.707500
3        C0004      5354.88              23          240.636250
4        C0005      2034.24               7          291.603333
..         …           …              …               …
194      C0196      4982.88              12          416.992500
195      C0197      1928.65               9          227.056667
196      C0198       931.83               3          239.705000
197      C0199      1979.28               9          250.610000
198      C0200      4758.60              16          296.506000

     num_transactions  Region_Europe  Region_North America  \
0                   5          False                 False
1                   4          False                 False
2                   4          False                 False
3                   8          False                 False
4                   3          False                 False
..                …            …                   …
194                 4           True                 False
195                 3           True                 False
196                 2           True                 False
197                 4           True                 False
198                 5          False                 False

     Region_South America
0                    True
1                   False
2                    True
3                    True
4                   False
..                    …
194                 False
195                 False
196                 False
197                 False
198                 False
```

```
[199 rows x 8 columns]
```

[9]:
```python
# Standardize numeric features
numeric_columns = ['total_spent', 'total_quantity', 'avg_price_per_item',␣
↪'num_transactions']
scaler = StandardScaler()
customer_features_encoded[numeric_columns] = scaler.
↪fit_transform(customer_features_encoded[numeric_columns])

customer_features_encoded
```

[9]:
```
    CustomerID  total_spent  total_quantity  avg_price_per_item  \
0        C0001    -0.061701       -0.122033            0.094670
1        C0002    -0.877744       -0.448000           -0.904016
2        C0003    -0.405857        0.203934           -1.094109
3        C0004     1.032547        1.670787           -0.447702
4        C0005    -0.783929       -0.936951            0.285581
..         …           …             …                 …
194      C0196     0.829053       -0.122033            2.089604
195      C0197    -0.841689       -0.610984           -0.643077
196      C0198    -1.386975       -1.588886           -0.461100
197      C0199    -0.813993       -0.610984           -0.304206
198      C0200     0.706367        0.529902            0.356118

     num_transactions  Region_Europe  Region_North America  \
0           -0.011458          False                 False
1           -0.467494          False                 False
2           -0.467494          False                 False
3            1.356650          False                 False
4           -0.923530          False                 False
..                …            …                   …
194         -0.467494           True                 False
195         -0.923530           True                 False
196         -1.379566           True                 False
197         -0.467494           True                 False
198         -0.011458          False                 False

     Region_South America
0                    True
1                   False
2                    True
3                    True
4                   False
..                    …
194                 False
195                 False
```

```
196                 False
197                 False
198                 False

[199 rows x 8 columns]
```

[10]:
```
# Step 2: Calculate Similarity Matrix
# Prepare data for similarity calculations
features = customer_features_encoded.drop(columns=['CustomerID'])
similarity_matrix = cosine_similarity(features)
similarity_matrix
```

[10]:
```
array([[ 1.        ,  0.0199149 ,  0.54942724, …,  0.09024666,
         0.06525033, -0.07707489],
       [ 0.0199149 ,  1.        ,  0.64192372, …,  0.76766827,
         0.68274041, -0.87028969],
       [ 0.54942724,  0.64192372,  1.        , …,  0.3117456 ,
         0.30522223, -0.36507641],
       …,
       [ 0.09024666,  0.76766827,  0.3117456 , …,  1.        ,
         0.92205414, -0.75226462],
       [ 0.06525033,  0.68274041,  0.30522223, …,  0.92205414,
         1.        , -0.68668544],
       [-0.07707489, -0.87028969, -0.36507641, …, -0.75226462,
        -0.68668544,  1.        ]])
```

[11]:
```
# Convert similarity matrix to a DataFrame
similarity_df = pd.DataFrame(similarity_matrix,␣
 ↪index=customer_features_encoded['CustomerID'],␣
 ↪columns=customer_features_encoded['CustomerID'])
similarity_df
```

[11]:
```
CustomerID     C0001     C0002     C0003     C0004     C0005     C0006  \
CustomerID
C0001       1.000000  0.019915  0.549427  0.253296  0.126841  0.722861
C0002       0.019915  1.000000  0.641924 -0.506416  0.580630 -0.420805
C0003       0.549427  0.641924  1.000000  0.182760  0.097670  0.031933
C0004       0.253296 -0.506416  0.182760  1.000000 -0.917866  0.052609
C0005       0.126841  0.580630  0.097670 -0.917866  1.000000  0.204362
…                …         …         …         …         …         …
C0196       0.065663 -0.659737 -0.598273 -0.139329  0.126188  0.648900
C0197       0.041074  0.781417  0.456427 -0.594026  0.667077 -0.149335
C0198       0.090247  0.767668  0.311746 -0.796264  0.867865 -0.034124
C0199       0.065250  0.682740  0.305222 -0.586348  0.651741 -0.137393
C0200      -0.077075 -0.870290 -0.365076  0.575663 -0.632300  0.388358

CustomerID     C0007     C0008     C0009     C0010  …     C0191     C0192  \
```

```
CustomerID                                               …
C0001        0.138612 -0.087682  0.126885 -0.036006  …  0.970602  0.736231
C0002        0.076637 -0.359682  0.560566  0.833336  …  0.136618  0.473582
C0003       -0.273396 -0.078039  0.061060  0.672174  …  0.496668  0.452360
C0004       -0.786507  0.772672 -0.825994 -0.180041  …  0.059562 -0.440346
C0005        0.851398 -0.838844  0.894675  0.231137  …  0.311495  0.745070
…                 …         …         …         …  …       …         …
C0196        0.574539 -0.289118  0.199160 -0.553793  …  0.062902  0.028463
C0197        0.321154 -0.540695  0.804871  0.828388  …  0.148806  0.494528
C0198        0.562142 -0.684219  0.936408  0.639150  …  0.251288  0.660241
C0199        0.363455 -0.450854  0.877385  0.752139  …  0.203871  0.521984
C0200       -0.208591  0.259771 -0.694121 -0.627038  … -0.265237 -0.584790

CustomerID      C0193     C0194     C0195     C0196     C0197     C0198  \
CustomerID
C0001        0.071388 -0.057792  0.574169  0.065663  0.041074  0.090247
C0002        0.774186 -0.184328 -0.143869 -0.659737  0.781417  0.767668
C0003        0.393936 -0.028563  0.603883 -0.598273  0.456427  0.311746
C0004       -0.816679  0.521400  0.879219 -0.139329 -0.594026 -0.796264
C0005        0.901178 -0.584876 -0.650627  0.126188  0.667077  0.867865
…                 …         …         …         …         …         …
C0196       -0.130853 -0.288606 -0.327782  1.000000 -0.116666 -0.039813
C0197        0.795125 -0.346467 -0.281280 -0.116666  1.000000  0.935675
C0198        0.888703 -0.433573 -0.477282 -0.039813  0.935675  1.000000
C0199        0.672928 -0.280217 -0.324723 -0.004530  0.958107  0.922054
C0200       -0.624243  0.089755  0.281141  0.532105 -0.652410 -0.752265

CustomerID      C0199     C0200
CustomerID
C0001        0.065250 -0.077075
C0002        0.682740 -0.870290
C0003        0.305222 -0.365076
C0004       -0.586348  0.575663
C0005        0.651741 -0.632300
…                 …         …
C0196       -0.004530  0.532105
C0197        0.958107 -0.652410
C0198        0.922054 -0.752265
C0199        1.000000 -0.686685
C0200       -0.686685  1.000000

[199 rows x 199 columns]
```

[12]:
```python
# Step 3: Generate Lookalike Map
lookalike_map = {}
```

```
for customer_id in customer_features_encoded['CustomerID'][:20]:  # For␣
↪CustomerID C0001 to C0020
    # Get similarity scores for the current customer
    similar_customers = similarity_df[customer_id].sort_values(ascending=False)
    similar_customers = similar_customers.drop(index=customer_id)  # Exclude␣
↪the customer themselves
    top_3 = similar_customers.head(3)
    lookalike_map[customer_id] = [(similar_id, round(score, 4)) for similar_id,␣
↪score in top_3.items()]
lookalike_map
```

[12]:

```
{'C0001': [('C0137', 0.979), ('C0191', 0.9706), ('C0011', 0.9489)],
 'C0002': [('C0043', 0.9751), ('C0142', 0.9699), ('C0027', 0.9624)],
 'C0003': [('C0190', 0.9415), ('C0174', 0.8949), ('C0025', 0.8367)],
 'C0004': [('C0113', 0.9864), ('C0165', 0.9838), ('C0012', 0.9647)],
 'C0005': [('C0128', 0.996), ('C0123', 0.9931), ('C0140', 0.9909)],
 'C0006': [('C0168', 0.9339), ('C0048', 0.9243), ('C0158', 0.915)],
 'C0007': [('C0078', 0.9875), ('C0092', 0.9831), ('C0146', 0.9715)],
 'C0008': [('C0109', 0.9295), ('C0084', 0.9265), ('C0090', 0.9174)],
 'C0009': [('C0061', 0.9684), ('C0198', 0.9364), ('C0167', 0.935)],
 'C0010': [('C0121', 0.9711), ('C0111', 0.9349), ('C0060', 0.9034)],
 'C0011': [('C0107', 0.9758), ('C0001', 0.9489), ('C0137', 0.9307)],
 'C0012': [('C0102', 0.978), ('C0153', 0.9717), ('C0108', 0.9657)],
 'C0013': [('C0104', 0.9885), ('C0188', 0.9869), ('C0099', 0.9839)],
 'C0014': [('C0063', 0.9875), ('C0062', 0.9684), ('C0097', 0.9589)],
 'C0015': [('C0058', 0.9949), ('C0131', 0.9792), ('C0020', 0.959)],
 'C0016': [('C0079', 0.9337), ('C0185', 0.895), ('C0050', 0.8946)],
 'C0017': [('C0075', 0.9748), ('C0124', 0.9637), ('C0041', 0.9187)],
 'C0018': [('C0046', 0.8398), ('C0068', 0.8213), ('C0122', 0.7907)],
 'C0019': [('C0172', 0.9585), ('C0086', 0.887), ('C0119', 0.8606)],
 'C0020': [('C0131', 0.9636), ('C0015', 0.959), ('C0058', 0.9433)]}
```

[13]:

```
# Step 4: Output Results
# Save lookalike map to Lookalike.csv
lookalike_df = pd.DataFrame({
    'cust_id': lookalike_map.keys(),
    'similar_customers': [str(value) for value in lookalike_map.values()]
})

lookalike_df
```

[13]:

```
  cust_id                                  similar_customers
0   C0001  [('C0137', 0.979), ('C0191', 0.9706), ('C0011'…
1   C0002  [('C0043', 0.9751), ('C0142', 0.9699), ('C0027…
2   C0003  [('C0190', 0.9415), ('C0174', 0.8949), ('C0025…
3   C0004  [('C0113', 0.9864), ('C0165', 0.9838), ('C0012…
4   C0005  [('C0128', 0.996), ('C0123', 0.9931), ('C0140'…
```

```
5    C0006  [('C0168', 0.9339), ('C0048', 0.9243), ('C0158…
6    C0007  [('C0078', 0.9875), ('C0092', 0.9831), ('C0146…
7    C0008  [('C0109', 0.9295), ('C0084', 0.9265), ('C0090…
8    C0009  [('C0061', 0.9684), ('C0198', 0.9364), ('C0167…
9    C0010  [('C0121', 0.9711), ('C0111', 0.9349), ('C0060…
10   C0011  [('C0107', 0.9758), ('C0001', 0.9489), ('C0137…
11   C0012  [('C0102', 0.978), ('C0153', 0.9717), ('C0108'…
12   C0013  [('C0104', 0.9885), ('C0188', 0.9869), ('C0099…
13   C0014  [('C0063', 0.9875), ('C0062', 0.9684), ('C0097…
14   C0015  [('C0058', 0.9949), ('C0131', 0.9792), ('C0020…
15   C0016  [('C0079', 0.9337), ('C0185', 0.895), ('C0050'…
16   C0017  [('C0075', 0.9748), ('C0124', 0.9637), ('C0041…
17   C0018  [('C0046', 0.8398), ('C0068', 0.8213), ('C0122…
18   C0019  [('C0172', 0.9585), ('C0086', 0.887), ('C0119'…
19   C0020  [('C0131', 0.9636), ('C0015', 0.959), ('C0058'…
```

[14]:
```
lookalike_df.to_csv("Dnyanesh_Shinde_Lookalike.csv", index=False)

print("Dnyanesh_Shinde_Lookalike.csv has been generated successfully!")
```

```
Dnyanesh_Shinde_Lookalike.csv has been generated successfully!
```