task

November 19, 2024

[1]:
```
import pandas as pd
```

[2]:
```
df  = pd.read_csv("retail_sales_dataset.csv")
```

[3]:
```
df
```

[3]:
```
     Transaction ID        Date Customer ID  Gender  Age Product Category  \
0                 1  2023-11-24     CUST001    Male   34           Beauty
1                 2  2023-02-27     CUST002  Female   26         Clothing
2                 3  2023-01-13     CUST003    Male   50      Electronics
3                 4  2023-05-21     CUST004    Male   37         Clothing
4                 5  2023-05-06     CUST005    Male   30           Beauty
..              ...         ...         ...     ...  ...              ...
995             996  2023-05-16     CUST996    Male   62         Clothing
996             997  2023-11-17     CUST997    Male   52           Beauty
997             998  2023-10-29     CUST998  Female   23           Beauty
998             999  2023-12-05     CUST999  Female   36      Electronics
999            1000  2023-04-12    CUST1000    Male   47      Electronics

     Quantity  Price per Unit  Total Amount
0           3              50           150
1           2             500          1000
2           1              30            30
3           1             500           500
4           2              50           100
..        ...             ...           ...
995         1              50            50
996         3              30            90
997         4              25           100
998         3              50           150
999         4              30           120

[1000 rows x 9 columns]
```

[5]:
```
df.tail()
```

[5]:
```
     Transaction ID        Date Customer ID  Gender  Age Product Category  \
995             996  2023-05-16     CUST996    Male   62         Clothing
996             997  2023-11-17     CUST997    Male   52           Beauty
997             998  2023-10-29     CUST998  Female   23           Beauty
998             999  2023-12-05     CUST999  Female   36      Electronics
999            1000  2023-04-12    CUST1000    Male   47      Electronics

     Quantity  Price per Unit  Total Amount
995         1              50            50
996         3              30            90
997         4              25           100
998         3              50           150
999         4              30           120
```

[6]:
```
df.head()
```

[6]:
```
   Transaction ID        Date Customer ID  Gender  Age Product Category  \
0               1  2023-11-24     CUST001    Male   34           Beauty
1               2  2023-02-27     CUST002  Female   26         Clothing
2               3  2023-01-13     CUST003    Male   50      Electronics
3               4  2023-05-21     CUST004    Male   37         Clothing
4               5  2023-05-06     CUST005    Male   30           Beauty

   Quantity  Price per Unit  Total Amount
0         3              50           150
1         2             500          1000
2         1              30            30
3         1             500           500
4         2              50           100
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    1000 non-null   int64
 1   Date              1000 non-null   object
 2   Customer ID       1000 non-null   object
 3   Gender            1000 non-null   object
 4   Age               1000 non-null   int64
 5   Product Category  1000 non-null   object
 6   Quantity          1000 non-null   int64
 7   Price per Unit    1000 non-null   int64
 8   Total Amount      1000 non-null   int64
dtypes: int64(5), object(4)
```

```
memory usage: 70.4+ KB
```

[8]:
```
df.describe()
```

[8]:
```
       Transaction ID         Age     Quantity  Price per Unit  Total Amount
count     1000.000000  1000.00000  1000.000000     1000.000000   1000.000000
mean       500.500000    41.39200     2.514000      179.890000    456.000000
std        288.819436    13.68143     1.132734      189.681356    559.997632
min          1.000000    18.00000     1.000000       25.000000     25.000000
25%        250.750000    29.00000     1.000000       30.000000     60.000000
50%        500.500000    42.00000     3.000000       50.000000    135.000000
75%        750.250000    53.00000     4.000000      300.000000    900.000000
max       1000.000000    64.00000     4.000000      500.000000   2000.000000
```

[9]:
```
df.columns
```

[9]:
```
Index(['Transaction ID', 'Date', 'Customer ID', 'Gender', 'Age',
       'Product Category', 'Quantity', 'Price per Unit', 'Total Amount'],
      dtype='object')
```

[10]:
```
df.dtypes
```

[10]:
```
Transaction ID      int64
Date               object
Customer ID        object
Gender             object
Age                 int64
Product Category   object
Quantity            int64
Price per Unit      int64
Total Amount        int64
dtype: object
```

[11]:
```
df.isnull().sum()
```

[11]:
```
Transaction ID     0
Date               0
Customer ID        0
Gender             0
Age                0
Product Category   0
Quantity           0
Price per Unit     0
Total Amount       0
dtype: int64
```

[15]:
```
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.impute import SimpleImputer
```

[19]:
```
user_product_matrix = df.pivot_table(
index = 'Customer ID',
columns='Product Category',
values = 'Total Amount',
aggfunc = 'sum'
).fillna(0)
```

[22]:
```
user_similarity = cosine_similarity(user_product_matrix)
user_similarity_df = pd.DataFrame(user_similarity,index=user_product_matrix.
 ↪index,columns=user_product_matrix.index)
```

[23]:
```
# Function to get recommendations for a given user
def recommend_products(user_id, n_recommendations=3):
    # Find the most similar users
    similar_users = user_similarity_df[user_id].sort_values(ascending=False)[1:
 ↪]  # Exclude the user itself
    similar_user_id = similar_users.index[0]

    # Get products purchased by the similar user
    similar_user_products = user_product_matrix.loc[similar_user_id]
    user_products = user_product_matrix.loc[user_id]

    # Recommend products not yet purchased by the user
    recommendations = similar_user_products[user_products == 0]
    recommendations = recommendations.sort_values(ascending=False).
 ↪head(n_recommendations)

    return recommendations.index.tolist()

# Example usage
recommendations = recommend_products('CUST001')
print(f"Recommendations for CUST001: {recommendations}")
```

```
Recommendations for CUST001: ['Clothing', 'Electronics']
```

[26]:
```
from sklearn.preprocessing import OneHotEncoder

# One-hot encode the product category
encoder = OneHotEncoder()
category_encoded = encoder.fit_transform(df[['Product Category']])

# Add price-related features
```

```
product_features = np.hstack((category_encoded.toarray(), df[['Price per Unit',␣
 ↪'Quantity']].values))
```

[33]:
```
# Aggregate data by product category
category_features = df.groupby('Product Category')[['Price per Unit',␣
 ↪'Quantity']].mean()

# If you have one-hot encoded categories, take their mean too
category_one_hot = pd.get_dummies(df['Product Category'])
category_features = pd.concat([category_features, category_one_hot.mean()],␣
 ↪axis=1)

print(category_features.head())
```

```
             Price per Unit  Quantity      0
Beauty           184.055375  2.511401  0.307
Clothing         174.287749  2.547009  0.351
Electronics      181.900585  2.482456  0.342
```

[34]:
```
# Compute similarity based on aggregated features
product_similarity = cosine_similarity(category_features)

# Create a DataFrame for the similarity matrix
product_similarity_df = pd.DataFrame(
    product_similarity,
    index=category_features.index,
    columns=category_features.index
)

print(product_similarity_df)
```

```
                Beauty  Clothing  Electronics
Beauty        1.000000  0.999999          1.0
Clothing      0.999999  1.000000          1.0
Electronics   1.000000  1.000000          1.0
```

[35]:
```
# Function to recommend similar products
def recommend_similar_products(product_category, n_recommendations=3):
    similar_products = product_similarity_df[product_category].
 ↪sort_values(ascending=False)[1:]  # Exclude itself
    return similar_products.head(n_recommendations).index.tolist()

# Example usage
similar_products = recommend_similar_products('Clothing')
print(f"Products similar to 'Clothing': {similar_products}")
```

```
Products similar to 'Clothing': ['Electronics', 'Beauty']
```

[36]:
```python
from sklearn.metrics import precision_score, recall_score, f1_score

# Sample evaluation (requires ground truth and predictions)
y_true = [1, 1, 0, 0, 1]  # Ground truth labels
y_pred = [1, 0, 0, 1, 1]  # Predicted labels

precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)

print(f"Precision: {precision}, Recall: {recall}, F1-Score: {f1}")
```

```
Precision: 0.6666666666666666, Recall: 0.6666666666666666, F1-Score:
0.6666666666666666
```

[ ]:

[ ]: