Lookalike Model Documentation

Title: Lookalike Model Documentation for eCommerce Transactions Dataset

Prepared by: Atul Kumar Suthar

Date: 26-jan-2025

Project: eCommerce Transactions Dataset Analysis

Overview

This script implements a Lookalike Model that identifies the top 3 similar customers for each customer in the dataset based on their transaction and product purchase behavior. The similarity is calculated using cosine similarity between customer profiles.

Steps in the Script

1. Load Datasets

The script loads three datasets:

- **Customers.csv**: Contains customer information.
- **Products.csv**: Contains product details.
- **Transactions.csv**: Contains transaction data.

Code Snippet:

```
customers = pd.read_csv("Customers.csv")

products = pd.read_csv("Products.csv")

transactions = pd.read_csv("Transactions.csv")
```

---

2. Merge Datasets

- Transactions are merged with products on ProductID to include product information in the transaction data.

- A fallback mechanism is implemented to handle cases where the Price column is missing in the Products.csv dataset.

Code Snippet:

```
merged = pd.merge(transactions, products, on="ProductID", how="left")


if 'Price' not in merged.columns:
    print("Warning: 'Price' column is missing. Creating a proxy price from TotalValue and Quantity.")
    merged['Price'] = merged['TotalValue'] / merged['Quantity']
    merged['Price'] = merged['Price'].fillna(0)
```

Then, the merged dataset is further merged with the customers dataset on CustomerID to include customer information.

Code Snippet:

```
merged = pd.merge(merged, customers, on="CustomerID", how="left")
```

---

3. Create Customer Profiles

- A summary profile is created for each customer by aggregating their transaction data:
    - **TotalValue**: Sum of total transaction values.
    - **Quantity**: Total quantity purchased.
    - **Price**: Average price of purchased products.

Code Snippet:

```
customer_profiles = merged.groupby("CustomerID").agg({
    'TotalValue': 'sum',  # Total transaction value
    'Quantity': 'sum',    # Total quantity purchased
    'Price': 'mean'       # Average product price
}).reset_index()
```

---

4. Calculate Similarity

- Cosine similarity is computed between the numerical features of the customer profiles.
- CustomerID is excluded from the similarity computation as it is non-numerical.

Code Snippet:

```
numerical_features = customer_profiles.drop("CustomerID", axis=1)

similarity = cosine_similarity(numerical_features)
```

---

5. Generate Lookalike Recommendations

- For each customer, the top 3 most similar customers are identified based on their similarity scores.
- The recommendations are stored in a list, containing the CustomerID and their top 3 lookalikes with similarity scores rounded to 2 decimal places.

Code Snippet:

```
recommendations = []

for i, customer in enumerate(customer_profiles['CustomerID']):
  scores = list(enumerate(similarity[i]))
  scores = sorted(scores, key=lambda x: x[1], reverse=True)[1:4]  # Top 3 similar customers
  recommendations.append({
    'CustomerID': customer,
    'Lookalikes': [(customer_profiles['CustomerID'][j], round(score, 2)) for j, score in scores]
  })
```

---

6. Save Recommendations to CSV

- The recommendations are written to a CSV file named Lookalike.csv.
- Each row contains a CustomerID and their top 3 lookalikes with similarity scores.

Code Snippet:

```
with open('Lookalike.csv', 'w', newline='') as f:
  writer = csv.writer(f)
  writer.writerow(['CustomerID', 'Lookalikes'])
  for r in recommendations:
    lookalikes_str = '; '.join([f"({cust_id}, {score:.2f})" for cust_id, score in r['Lookalikes']])
    writer.writerow([r['CustomerID'], lookalikes_str])
```

Output

- A CSV file named Lookalike.csv is created.
- Each row contains:
    - **CustomerID**: The customer for whom lookalikes are generated.
    - **Lookalikes**: A list of tuples representing the top 3 similar customers and their similarity scores.

Sample Output (Lookalike.csv):

```
CustomerID,Lookalikes
C0001,"(C0002, 0.98); (C0003, 0.95); (C0004, 0.93)"
C0002,"(C0001, 0.98); (C0005, 0.94); (C0006, 0.92)"
C0003,"(C0001, 0.95); (C0004, 0.93); (C0007, 0.91)"
```

Features and Benefits

1. **Fallback for Missing Data**:
    - If the Price column is missing, it is calculated as TotalValue / Quantity to ensure the model works seamlessly.
2. **Customizable Similarity Logic**:
    - The model uses cosine similarity, but it can be extended to other similarity metrics.

3. **Readable Output**:
    - Recommendations are formatted as strings for easy interpretation.

How to Run

1. Place Customers.csv, Products.csv, and Transactions.csv in the same directory as the script.
2. Run the script:
3. python lookalike_model.py
4. Check the Lookalike.csv file for the output.