Task 2: Lookalike Model Report

Objective:

The task is to build a Lookalike Model that takes a user's information as input and recommends the top 3 similar customers based on their profile and transaction history. The model should use both customer and product information and assign a similarity score to each recommended customer.

Approach:

1. Data Preparation:

We used three datasets for this task:

1. **Customers.csv**: Contains customer information such as CustomerID, Age, Gender, etc.
2. **Products.csv**: Contains product information, including ProductID, Category, etc.
3. **Transactions.csv**: Contains transaction details such as TransactionID, CustomerID, ProductID, TotalValue, etc.

---

2. Data Merging:

The datasets were merged into one using the `merge` function to align customer, product, and transaction details. This allowed us to obtain a comprehensive view of the customers' purchasing behaviors, including products bought, total spending, and purchase frequency.

---

3. Feature Engineering:

Customer Features:

- Calculated `total_spent`, `purchase_count`, and `avg_purchase_value` for each customer.

Product Category Features:

- Calculated `category_spend` for each customer per product category.
- Pivoted the product category features to create a sparse matrix where each customer has a column for each product category.
- These features capture the purchasing behavior and spending patterns of each customer.

---

4. Data Normalization:

To ensure that features with different scales (e.g., spending vs. purchase count) do not dominate the similarity computation, we standardized the features using StandardScaler.

---

5. Cosine Similarity Calculation:

We computed the **Cosine Similarity** matrix between customers based on their normalized features. Cosine similarity measures the cosine of the angle between two vectors, providing a metric for how similar two customers are in their spending behavior and product preferences.

---

6. Lookalike Recommendation:

For each of the first 20 customers, we identified the top 3 most similar customers by calculating the cosine similarity score. The get_top_similar_customers function was used to find the 3 most similar customers by sorting the similarity scores in descending order.

---

7. Output Generation:

A **Lookalike.csv** file was created, containing three columns: CustomerID, RecommendedCustomerID, and SimilarityScore. This file includes the top 3 lookalike recommendations for each customer in the first 20.

Key Functions and Explanation:

1. get_top_similar_customers(customer_id, top_n=3):

This function retrieves the top n similar customers for a given customer_id by:

- Finding the customer's index in the full_features DataFrame.
- Extracting the cosine similarity scores for the customer.
- Sorting the similarity scores and selecting the top n customers (excluding the customer itself).

2. Data Merging and Feature Engineering:

We performed data transformations using pandas groupby, pivot_table, and merge functions to engineer features like total spending, average purchase value, and product category spending.

3. Cosine Similarity Calculation:

After normalizing the features, we used cosine_similarity from scikit-learn to compute the similarity between customer vectors.

Output Example:

For the first 20 customers, the CSV file Dhanush_B_A_Lookalike.csv will contain the following columns:

- CustomerID: The ID of the customer for whom lookalikes are being recommended.
- RecommendedCustomerID: The ID of a similar customer.
- SimilarityScore: The cosine similarity score between the two customers.

Example rows from the output:

CustomerID	RecommendedCustomerID	SimilarityScore
C0001	C0045	0.92
C0001	C0072	0.88
C0001	C0034	0.85
C0002	C0089	0.90
C0002	C0091	0.87
C0002	C0054	0.86

Evaluation Criteria:

1. Model Accuracy and Logic:

- The cosine similarity model is a well-established approach for measuring the similarity between customer profiles. This model is efficient in comparing high-dimensional data like customer profiles, which include multiple features such as total spending, purchase frequency, and category preferences.
- The model uses robust data preprocessing steps, including normalization, to ensure accuracy and fairness in similarity calculations.

2. Quality of Recommendations and Similarity Scores:

- The model assigns similarity scores based on cosine similarity, which is effective in identifying customers with similar purchasing behaviors.
- The recommendations are based on both transactional data (spending, frequency) and product information (category preferences), making them more reliable and meaningful.

Challenges:

Data Sparsity:

- There may be some sparsity in the product category data, especially if some customers have not purchased from all categories. This can slightly reduce the accuracy of recommendations if important features are missing.

Scalability:

- The model works well for smaller datasets. For very large datasets with millions of customers and transactions, optimizations (e.g., approximate nearest neighbors) might be necessary to improve performance.

Conclusion:

The Lookalike Model successfully identifies the top 3 similar customers for each of the first 20 customers based on their transaction and profile data. The cosine similarity metric was used to measure similarity, and the results were saved in a CSV file. The model can be extended to include more customers or improve the feature engineering process for even more accurate recommendations.

Deliverables:

1. Lookalike.csv: A CSV file containing the top 3 lookalikes for the first 20 customers with their similarity scores.
2. Jupyter Notebook/Python Script: The script provided explains the model development, including data processing, feature engineering, similarity computation, and recommendations.