Lookalike Model

Explanation of the Code and Process

This script processes customer, product, and transaction data to build a lookalike customer recommendation system using a combination of data aggregation, scaling, cosine similarity, and clustering techniques.

Steps in the Script

Import Required Libraries The script uses essential Python libraries for data manipulation and machine learning:

NumPy: For numerical computations.

Pandas: For reading CSV files and data manipulation.

Scikit-learn: For scaling, similarity computation, clustering, and evaluation.

Load Data Data is read from three CSV files:

Customers.csv

Products.csv

Transactions.csv

```
Cust_data = pd.read_csv('Customers.csv')

Prod_data = pd.read_csv('Products.csv')

Trans_data = pd.read_csv('Transactions.csv')
```

Merge Datasets The merge function is used to combine data from all three datasets:

Transactions and products are merged on ProductID.

The resulting dataset is then merged with customer data on CustomerID.

```
transactions = pd.merge(Trans_data, Prod_data, on="ProductID", how="left")

data = pd.merge(transactions, Cust_data, on="CustomerID", how="left")
```

Aggregate Customer Features Customer-level features are aggregated using groupby and .agg():

total_spent: Sum of TotalValue for each customer.

transaction_count: Total number of transactions for each customer.

distinct_products: Number of unique products purchased by each customer.

```
customer_features = data.groupby("CustomerID").agg(
    total_spent=("TotalValue", "sum"),
    transaction_count=("TransactionID", "count"),
    distinct_products=("ProductID", "nunique")
).reset_index()
```

Normalize Data The features are scaled to a uniform range (0 to 1) using MinMaxScaler to ensure compatibility during similarity computation.

```
scaler = MinMaxScaler()
customer_features_scaled = scaler.fit_transform(customer_features.iloc[:, 1:])
```

Compute Cosine Similarity The cosine similarity is calculated between customers based on their scaled features. This similarity matrix measures how similar each customer is to others.

```
similarity_matrix = cosine_similarity(customer_features_scaled)
similarity_df = pd.DataFrame(similarity_matrix, index=customer_ids,
columns=customer_ids)
```

Generate Lookalike Recommendations For the 20 customers in the dataset:

Find the top 3 most similar customers (excluding the customer itself).

Store the results as a dictionary, where keys are customer IDs, and values are lists of similar customers and their similarity scores.

```
lookalike_results = {}
for customer_id in similarity_df.index[:20]:
    similar_customers = similarity_df.loc[customer_id].sort_values(ascending=False)[1:4]
    lookalike_results[customer_id] = list(zip(similar_customers.index,
similar_customers.values))
```

Accessing Lookalike Results The dictionary lookalike_results contains the top similar customers for the 20 customers. For example, to find lookalikes for customer C0001:

```
lookalike_results['C0001']
```

Generate CSV file of 20 lookalike customers:

```
lookalike_dataframe = pd.DataFrame({
    'cust_id': lookalike_results.keys(),
    'lookalikes': [str(value) for value in lookalike_results.values()]  # Convert the list of tuples to strings
})


# Save to CSV
lookalike_dataframe.to_csv('Lookalike.csv', index=False)
```

This code creates csv file

Key Outputs

Customer Features:

Aggregated metrics such as total_spent, transaction_count, and distinct_products for each customer.

Scaled Features:

A normalized version of the customer features to ensure fair comparison.

Similarity Matrix:

A pairwise cosine similarity matrix showing similarity scores between all customers.

Lookalike Results:

A dictionary of similar customers for the given 200 customers, along with similarity scores.

Example Insights

The similarity score for customer C0001 and their top 3 similar customers is returned via `lookalike_results['C0001']`

These lookalike results can be used for:

Personalized marketing campaigns.

Recommendations for cross-selling or upselling products.

Enhancing customer engagement by targeting customers with similar spending patterns.

For new customers

```
def get_lookalike_customers(new_customer_features,old_customer_features):

    old_customer_features.loc[len(old_customer_features)]=new_customer_features

    customer_features=old_customer_features.copy()

    customer_ids=customer_features["CustomerID"]

    similarity_matrix = cosine_similarity(customer_features_scaled)

    similarity_df = pd.DataFrame(similarity_matrix, index=customer_ids,
columns=customer_ids)


    # Generate lookalike results

    lookalike_results = {}

    customer_id = similarity_df.index[-1]

    similar_customers = similarity_df.loc[customer_id].sort_values(ascending=False)[1:4]

    lookalike_results[customer_id] = list(zip(similar_customers.index,
similar_customers.values))

    return lookalike_results
```

this function takes the inputs as old customer features and new customer features and return lookalike result for new customer.

Example usage of the function:

```
lookalike=get_lookalike_customers(new_customer_features,old_customer_features)
```

Result will be stored in lookalike.