Here is a **Jupyter Notebook**/Python script that explains the model development for the lookalike recommendation task using customer data. The notebook walks through the steps of data preprocessing, similarity calculation, and generating lookalike recommendations.

```
# Importing necessary libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.metrics.pairwise import cosine_similarity


# Step 1: Load the customer and transaction data

# For this example, the 'Customers.csv' contains customer demographics

# 'Transactions.csv' contains transaction data for each customer

customers = pd.read_csv('Customers.csv', encoding='utf-8')

transactions = pd.read_csv('Transactions.csv', encoding='utf-8')


# Step 2: Data Cleaning and Feature Engineering


# Assuming the 'Customers.csv' contains the following columns:

# CustomerID, Age, Gender, Location, etc.

# We will encode categorical features like 'Gender' and 'Location' using one-hot encoding

customers_encoded = pd.get_dummies(customers[['Gender', 'Location']])


# Now, combine customer demographic data with transaction features

# For simplicity, let's assume we aggregate transaction data by CustomerID and calculate total spending and product diversity

transaction_summary = transactions.groupby('CustomerID').agg({

    'Amount': 'sum',  # Total spending for each customer

    'ProductID': 'nunique',  # Number of unique products bought

}).reset_index()


# Step 3: Merge customer demographics with transaction summary
```

```
customer_profiles = pd.merge(customers[['CustomerID', 'Age']], transaction_summary, on='CustomerID')
customer_profiles = pd.concat([customer_profiles, customers_encoded], axis=1)

# Step 4: Standardizing the data
# Standardize the numerical data (Age, Amount, and ProductID) to avoid scale issues
scaler = StandardScaler()
scaled_profiles = scaler.fit_transform(customer_profiles.drop('CustomerID', axis=1))

# Step 5: Dimensionality Reduction using PCA (optional)
# PCA is used to reduce the number of features while preserving as much variance as possible
pca = PCA(n_components=5)  # We are reducing to 5 principal components
reduced_profiles = pca.fit_transform(scaled_profiles)

# Step 6: Calculate Cosine Similarity between customer profiles
# Cosine similarity is used to measure the similarity between customers based on their profiles
similarities = cosine_similarity(reduced_profiles)

# Step 7: Extract Top 3 Lookalikes for each customer (CustomerID: C0001 to C0020)
top_lookalikes = {}

for idx in range(20):  # Loop through customers C0001 to C0020 (first 20 rows)
    similarities_idx = similarities[idx]  # Get similarity scores for the current customer
    similar_customers = [(i, similarities_idx[i]) for i in range(len(similarities_idx)) if i != idx]
    similar_customers.sort(key=lambda x: x[1], reverse=True)  # Sort customers based on similarity score
    top_lookalikes[customers.iloc[idx]['CustomerID']] = [
        {'CustomerID': customers.iloc[i[0]]['CustomerID'], 'Score': i[1]} for i in similar_customers[:3]
    ]

# Step 8: Save the recommendations into 'Lookalike.csv' with the format: Map<cust_id, List<cust_id, score>>
```

```
lookalike_data = []

# Create the list of lookalike recommendations
for cust_id, lookalikes in top_lookalikes.items():
    for lookalike in lookalikes:
        lookalike_data.append([cust_id, lookalike['CustomerID'], lookalike['Score']])

# Convert the list into a DataFrame and save it to a CSV
lookalike_df = pd.DataFrame(lookalike_data, columns=['CustomerID', 'LookalikeID', 'Score'])
lookalike_df.to_csv('Lookalike.csv', index=False)

# Step 9: Display the top 3 lookalikes for each customer
lookalike_df.head()  # Display the first few rows of the resulting lookalikes CSV

# Print message indicating the CSV has been saved successfully
print("Lookalike recommendations have been saved to 'Lookalike.csv'.")
```

Explanation of the Steps in the Notebook

1. Data Loading

- The script loads Customers.csv (containing customer demographics) and Transactions.csv (containing transaction data). If you encounter issues reading the files due to encoding, consider adjusting the encoding as mentioned earlier (e.g., using 'utf-8', 'latin1', or 'unicode_escape').

2. Feature Engineering

- The customer demographic data (Gender, Location, Age) is processed:
    - **One-hot encoding** is applied to the categorical variables like Gender and Location.
    - **Transaction aggregation**: For each customer, we calculate:
        - Total spending (Amount).
        - Product diversity (ProductID).

3. Data Scaling

- The **StandardScaler** is used to scale numerical features (e.g., Age, Amount, ProductID) so that they are all on the same scale.

4. Dimensionality Reduction (PCA)

- **Principal Component Analysis (PCA)** is applied to reduce the feature set to the most important components, simplifying the feature space and improving model efficiency.

5. Cosine Similarity

- Cosine similarity is calculated between the customer profiles, which helps to measure how similar two customers are based on their transaction history and demographic features. The higher the similarity score (closer to 1), the more similar the customers are.

6. Top 3 Lookalike Selection

- For each of the first 20 customers (C0001 - C0020), the script computes the **top 3 most similar customers** based on their cosine similarity score.

7. Saving Results

- The recommendations are stored in Lookalike.csv with each row containing:
    - CustomerID: The original customer.
    - LookalikeID: The recommended similar customer.
    - Score: The similarity score between the two customers.

8. Displaying Results

- Finally, the script displays the first few rows of the Lookalike.csv to show the results of the lookalike recommendations.

Summary of Model Evaluation

- **Accuracy and Logic**:
    - The model uses **Cosine Similarity** to calculate how similar each customer is to others based on both their demographic and transactional data.
    - **PCA** ensures the model is not bogged down by high-dimensional data and focuses on the most important features.
    - The recommendations are made based on high similarity scores, ensuring that the lookalikes share meaningful similarities in behavior.
- **Output Quality**:
    - The output file, Lookalike.csv, contains the top 3 most similar customers for each of the first 20 customers, which can be used to improve customer targeting and personalization.

Running the Code

- To run the notebook, you should have access to the customer and transaction data in the correct CSV format.

- You can adjust the code to include additional customer features or modify how transactions are aggregated, depending on your dataset's structure.