Task 2: Lookalike Model

Approach:

1.Similarity Definition:

1.Use a combination of numerical features (e.g., TotalValue, Quantity) and categorical features (e.g., Region, Category).

2.Use cosine similarity or a distance metric (e.g., Euclidean distance).

2.Steps:

1. Preprocess the data (normalize numerical features, one-hot encode categorical features).

2.Create a feature matrix for customers using both profile and transaction data.

1. Compute similarity scores between customers.

3. Deliverables:

A CSV mapping customer IDs to their top 3 lookalikes and similarity scores.

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the combined dataset
combined_data = pd.read_csv('KishoreReddy_V_Combined_Data.csv')

# Feature matrix: Customer profile + transaction data
customer_features = combined_data.groupby('CustomerID').agg({
    'TotalValue': 'sum',
    'Quantity': 'sum',
    'Region': 'first'
}).reset_index()

customer_features
```

```
    CustomerID  TotalValue  Quantity         Region
0        C0001     3354.52        12  South America
1        C0002     1862.74        10           Asia
2        C0003     2725.38        14  South America
3        C0004     5354.88        23  South America
4        C0005     2034.24         7           Asia
..         ...         ...       ...            ...
194      C0196     4982.88        12         Europe
195      C0197     1928.65         9         Europe
```

```
196      C0198      931.83         3         Europe
197      C0199     1979.28         9         Europe
198      C0200     4758.60        16           Asia

[199 rows x 4 columns]

# One-hot encode categorical data
encoder = OneHotEncoder()
region_encoded =
encoder.fit_transform(customer_features[['Region']]).toarray()

# Normalize numerical data
scaler = StandardScaler()
numerical_features =
scaler.fit_transform(customer_features[['TotalValue', 'Quantity']])

# Combine features
features = np.hstack((numerical_features, region_encoded))

# Compute similarity
similarity_matrix = cosine_similarity(features)

# Get top 3 lookalikes
lookalike_results = {}
for i, customer_id in enumerate(customer_features['CustomerID']):
    similar_indices = np.argsort(-similarity_matrix[i])[:4]  # Top 3 +
itself
    similar_customers = [(customer_features['CustomerID'][j],
similarity_matrix[i][j])
                         for j in similar_indices if j != i]
    lookalike_results[customer_id] = similar_customers[:3]

# Save lookalikes to a CSV
lookalike_df = pd.DataFrame([
    {'CustomerID': cust, 'Lookalikes': lookalikes}
    for cust, lookalikes in lookalike_results.items()
])
lookalike_df.to_csv('Lookalike.csv', index=False)

lookalike = pd.read_csv("Lookalike.csv")

lookalike.head()

  CustomerID                                          Lookalikes
0      C0001  [('C0107', 0.9893604766330638), ('C0137', 0.98...
1      C0002  [('C0088', 0.9960799027166513), ('C0142', 0.98...
2      C0003  [('C0147', 0.9942553453615234), ('C0190', 0.99...
3      C0004  [('C0113', 0.9939871237922757), ('C0165', 0.98...
4      C0005  [('C0186', 0.9969474860655397), ('C0159', 0.99...
```

Task 3: Customer Segmentation/Clustering

Approach:

1. Clustering Features:
    1. Use customer profile and transaction data for clustering.
    2. Select features like TotalValue, Quantity, and Region.

2. Algorithm:

1. Use K-Means or Hierarchical Clustering.
2. Evaluate clusters using the Davies-Bouldin (DB) Index.

3.Visualization:

Plot clusters in 2D/3D space using PCA or t-SNE.

```
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.decomposition import PCA

# Prepare clustering data
features = np.hstack((numerical_features, region_encoded))

# Apply K-Means clustering
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(features)

# Calculate DB Index
db_index = davies_bouldin_score(features, clusters)
print("Davies-Bouldin Index:", db_index)
```

```
Davies-Bouldin Index: 1.4451410863711218
```

```
# Visualize clusters using PCA
pca = PCA(n_components=2)
pca_features = pca.fit_transform(features)
sns.scatterplot(x=pca_features[:, 0], y=pca_features[:, 1],
hue=clusters, palette='viridis')
plt.title('Customer Clusters')
plt.show()
```


Customer Clusters
0
1
2
3
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−2
−1
0
1
2
3
4