# Task-2 :

For **Task 2: Lookalike Model**, the evaluation criteria are **Model Accuracy and Logic** and **Quality of Recommendations and Similarity Scores**. Below, I'll break down how to address these criteria and ensure your model meets the expectations.

---

**1. Model Accuracy and Logic**

**What is Being Evaluated?**

- The logic behind how the model calculates similarity between customers.
- The accuracy of the similarity scores and recommendations.

**How to Ensure Good Model Accuracy and Logic:**

1. **Feature Engineering**:
    - Use meaningful features derived from customer profiles and transaction history. For example:
        - **Total Spending**: Total amount spent by the customer.
        - **Average Order Value**: Average amount spent per transaction.
        - **Favorite Category**: The most frequently purchased product category.
        - **Region**: Geographic location of the customer.
        - **Signup Date**: How long the customer has been active.
    - Ensure that the features are normalized or standardized to avoid bias due to different scales.
2. **Similarity Metric**:
    - Use **cosine similarity** or **Euclidean distance** to calculate similarity between customers based on their feature vectors.
    - Cosine similarity is preferred for high-dimensional data and when the magnitude of the vectors is not important.
3. **Model Logic**:
    - The model should compare each customer's feature vector with all other customers and rank them based on similarity scores.
    - Ensure that the logic is transparent and well-documented in the code.
4. **Validation**:
    - Manually validate the recommendations for a few customers to ensure they make sense. For example:

        - If Customer A has a high similarity score with Customer B, check if they share similar purchasing behavior, favorite categories, or regions.

---

**2. Quality of Recommendations and Similarity Scores**

**What is Being Evaluated?**

- The relevance of the recommended customers.
- The consistency and interpretability of the similarity scores.

**How to Ensure High-Quality Recommendations:**

1. **Top 3 Recommendations**:
    - For each customer, recommend the top 3 most similar customers based on the similarity score.
    - Ensure that the recommendations are diverse and not just based on a single feature (e.g., region or total spending).
2. **Similarity Scores**:
    - The similarity scores should be consistent and interpretable. For example:
        - A score of 0.9 indicates very high similarity, while a score of 0.2 indicates low similarity.
    - Avoid recommending customers with very low similarity scores (e.g., below 0.5).
3. **Business Relevance**:
    - The recommendations should make sense from a business perspective. For example:
        - If Customer A is a high-spending customer, the lookalikes should also be high-spending customers.
        - If Customer B frequently buys electronics, the lookalikes should also have a preference for electronics.
4. **Visualization**:
    - Visualize the similarity scores and recommendations to ensure they are meaningful. For example:
        - Create a heatmap of the similarity matrix to see how customers are clustered.
        - Plot the top 3 recommendations for a few customers to validate their relevance.

[16]:

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import StandardScaler
# Feature Engineering
customer_features = merged_data.groupby('CustomerID').agg({


    'TotalValue': ['sum', 'mean'],
    'Quantity': ['sum', 'mean'],
    'Category': lambda x: x.mode()[0]  # Most frequent category
}).reset_index()

customer_features.columns = ['CustomerID', 'TotalSpending', 'AvgOrderValue', 'TotalQuantity', 'AvgQuantity', 'FavoriteCategory']
customer_features = pd.merge(customer_features, customers, on='CustomerID')
customer_features = pd.get_dummies(customer_features, columns=['Region', 'FavoriteCategory'])
scaler = StandardScaler()
scaled_features = scaler.fit_transform(customer_features.drop(columns=['CustomerID', 'CustomerName', 'SignupDate']))
# Calculate similarity matrix
similarity_matrix = cosine_similarity(scaled_features)
# Function to get top 3 lookalikes
def get_lookalikes(customer_index, similarity_matrix, top_n=3):
    similar_customers = similarity_matrix[customer_index]
    top_indices = similar_customers.argsort()[-top_n-1:-1][::-1]
    return [(customer_features.iloc[i]['CustomerID'], similar_customers[i]) for i in top_indices]
# Generate lookalikes for the first 20 customers
lookalike_results = {}
for i in range(20):
    customer_id = customer_features.iloc[i]['CustomerID']
    lookalike_results[customer_id] = get_lookalikes(i, similarity_matrix)
# Save results to CSV
lookalike_df = pd.DataFrame(lookalike_results).T.reset_index()
lookalike_df.columns = ['CustomerID', 'Lookalike1', 'Lookalike2', 'Lookalike3']
lookalike_df.to_csv('Lookalike.csv', index=False)
print(lookalike_df)
```

| | CustomerID | Lookalike1 | Lookalike2 | Lookalike3 |
|---|---|---|---|---|
| 0 | C0001 | (C0048, 0.9843820492385122) | (C0181, 0.9693125194066259) | (C0190, 0.9534908488044361) |
| 1 | C0002 | (C0088, 0.9638642709548509) | (C0106, 0.9593287450084653) | (C0056, 0.9448750465323151) |
| 2 | C0003 | (C0195, 0.8990868232751801) | (C0052, 0.8227483942774672) | (C0151, 0.797171898356219) |
| 3 | C0004 | (C0165, 0.9558572139355662) | (C0169, 0.9390884052790256) | (C0153, 0.9369436540306584) |
| 4 | C0005 | (C0186, 0.9924553804892453) | (C0146, 0.9773165757486251) | (C0007, 0.9501438612455692) |
| 5 | C0006 | (C0171, 0.974078345858161) | (C0168, 0.9733141325378166) | (C0153, 0.8823178362241069) |
| 6 | C0007 | (C0140, 0.9781779331673608) | (C0115, 0.9709648587361617) | (C0005, 0.9501438612455692) |
| 7 | C0008 | (C0189, 0.8649924629195013) | (C0038, 0.857493057016355) | (C0160, 0.8321817717190918) |
| 8 | C0009 | (C0198, 0.9786385778435766) | (C0103, 0.9641913844451484) | (C0061, 0.8517420047693066) |
| 9 | C0010 | (C0111, 0.9759472769666288) | (C0062, 0.9536136587201031) | (C0061, 0.7492527143714895) |
| 10 | C0011 | (C0126, 0.9402415462952101) | (C0137, 0.9396244232766903) | (C0153, 0.9361199750113831) |
| 11 | C0012 | (C0104, 0.9745028319134471) | (C0113, 0.969636506250873) | (C0195, 0.9669138567928879) |
| 12 | C0013 | (C0099, 0.987385972692313) | (C0108, 0.9425438438913867) | (C0107, 0.8493293619866377) |
| 13 | C0014 | (C0060, 0.9814367033020084) | (C0172, 0.9099212543481435) | (C0164, 0.8376471939840633) |
| 14 | C0015 | (C0131, 0.9687632381579074) | (C0036, 0.9126296913765485) | (C0094, 0.903577090392509) |
| 15 | C0016 | (C0183, 0.9693792731049902) | (C0067, 0.9521705892704551) | (C0042, 0.9518825893794081) |
| 16 | C0017 | (C0057, 0.9367907248292845) | (C0075, 0.9333417280809972) | (C0081, 0.8852170735476687) |
| 17 | C0018 | (C0117, 0.9107029543056708) | (C0046, 0.874190527279122) | (C0185, 0.8250255793045433) |
| 18 | C0019 | (C0121, 0.9647740462173685) | (C0081, 0.8792364639432088) | (C0132, 0.8174497597370606) |
| 19 | C0020 | (C0050, 0.9106752005466154) | (C0035, 0.847633520335308) | (C0026, 0.8472433765776954) |