Customer Segmentation Preprocessing Recommendations

Current Dataset Overview

Based on the final customer dataset (customer_data_final.csv) with 117,604 rows and 10 columns, I've analyzed the current state and recommend additional preprocessing steps to enhance your customer segmentation analysis.

Current Columns:

- order_id
- order_item_id
- product_id
- payment_value
- customer_id
- customer_unique_id
- customer_zip_code_prefix
- customer_city
- customer_state
- product_category_name_english

Recommended Preprocessing Steps

1. Aggregate to Customer Level

Your current dataset is at the order-item level (multiple rows per customer). For customer segmentation, you need one row per customer:

```python
# Group by customer_unique_id to get one row per customer
customer_level = df.groupby('customer_unique_id').agg({
    'order_id': 'nunique',          # Number of orders (frequency)
    'payment_value': 'sum',         # Total spend (monetary)
    'customer_state': 'first',      # Geographic info
    'customer_city': 'first',
    'customer_zip_code_prefix': 'first'
}).reset_index()
```

2. Create RFM Features (Recency, Frequency, Monetary)

RFM analysis is fundamental for e-commerce customer segmentation:

```python
# Frequency - already calculated as number of orders above
customer_level.rename(columns={'order_id': 'frequency'}, inplace=True)

# Monetary - total customer spend
# Already calculated as sum of payment_value above

# Add average order value
customer_level['avg_order_value'] = customer_level['payment_value'] / customer_level['freque

# Recency requires order dates which appear to be missing from your current dataset
# If available in the original orders dataset, add:
# recency_days = days since customer's last purchase
```

3. Add Product Category Preferences

Identify category affinities for each customer:

```python
# Get top categories
top_categories = df['product_category_name_english'].value_counts().nlargest(10).index

# Create category purchase counts per customer
category_counts = pd.crosstab(df['customer_unique_id'], df['product_category_name_english'])
category_counts = category_counts[top_categories]  # Keep only top categories

# Merge with customer level data
customer_segments = customer_level.merge(category_counts, on='customer_unique_id', how='left
customer_segments.fillna(0, inplace=True)
```

4. Add Customer Purchase Behavior Features

Create features describing purchase patterns:

```python
# Product diversity - number of different categories purchased
category_diversity = df.groupby('customer_unique_id')['product_category_name_english'].nuniq
customer_segments['category_diversity'] = category_diversity

# Items per order
items_per_order = df.groupby(['customer_unique_id', 'order_id'])['order_item_id'].count().re
avg_items = items_per_order.groupby('customer_unique_id')['order_item_id'].mean()
customer_segments['avg_items_per_order'] = avg_items
```

5. Feature Scaling

Before applying clustering algorithms, scale the features:

```python
from sklearn.preprocessing import StandardScaler

# Select numerical columns to scale
num_cols = ['frequency', 'payment_value', 'avg_order_value', 'category_diversity',
            'avg_items_per_order'] + list(top_categories)

scaler = StandardScaler()
customer_segments[num_cols] = scaler.fit_transform(customer_segments[num_cols])
```

Important Missing Features to Consider

1. **Recency information** - Without order dates, you're missing a crucial RFM component. If possible, merge with the orders dataset to get purchase timestamps.
2. **Customer account age** - How long customers have been buying from the store.
3. **Return behavior** - If available, information about product returns could be valuable.
4. **Seasonal buying patterns** - If timestamps are available, analysis of when customers tend to purchase.

Conclusion

With these preprocessing steps, your dataset will be well-prepared for customer segmentation. Focus particularly on:

1. Getting customer-level aggregation (one row per unique customer)
2. Creating complete RFM metrics

3. Adding category preference features
4. Scaling appropriately before clustering

These enhancements will provide a solid foundation for identifying meaningful customer segments that can drive targeted marketing strategies and business decisions.