

KDD Process Steps Summary



Amazon Customer Behavior Survey

Unable to display visualization

refer to this chat <https://chatgpt.com/share/66f35f69-58c4-8009-849e-f52819e98e66>

and let's start with the KDD process

We will use this dataset for KDD

About Dataset

- 1.age= age
- 2.gender= gender
- 3.Purchase_Frequency= How frequently do you make purchases on Amazon?
- 4.Purchase_Categories= What product categories do you typically purchase on Amazon?
- 5.Personalized_Recommendation_Frequency = Have you ever made a purchase based on personalized product recommendations from Amazon?
- 6.Browsing_Frequency =How often do you browse Amazon's website or app?
- 7.Product_Search_Method =How do you search for products on Amazon?
- 8.Search_Result_Exploration =Do you tend to explore multiple pages of search results or focus on the first page?
- 9.Customer_Reviews_Importance =How important are customer reviews in your decision-making process?
- 10.Add_to_Cart_Browsing =Do you add products to your cart while browsing on Amazon?
- 11.Cart_Completion_Frequency =How often do you complete the purchase after adding products to your cart?
- 12.Cart_Abandonment_Factors =What factors influence your decision to abandon a purchase in your cart?
- 13.Saveforlater_Frequency =Do you use Amazon's "Save for Later" feature, and if so, how often?
- 14.Review_Left =Have you ever left a product review on Amazon?
- 15.Review_Reliability =How much do you rely on product reviews when making a purchase?
- 16.Review_Helpfulness =Do you find helpful information from other customers' reviews?
- 17.Personalized_Recommendation_Frequency =How often do you receive personalized product recommendations from Amazon?
- 18.Recommendation_Helpfulness =Do you find the recommendations helpful?
- 19.Rating_Accuracy =How would you rate the relevance and accuracy of the recommendations you receive
- 20.Shopping_Satisfaction =How satisfied are you with your overall shopping experience on Amazon?
- 23.Service_Appreciation =What aspects of Amazon's services do you appreciate the most?
- 24.Improvement_Areas =Are there any areas where you think Amazon can improve?

go step by step, break it into chunk, ask me to continue after each step



Memory updated

Let's begin the KDD process with your Amazon customer behavior dataset. The KDD process can be broken down into several steps, starting with understanding the dataset and preparing the data for analysis.

Step 1: Data Selection

In this step, we determine which data will be used from the dataset and ensure that it is relevant to the task.

I'll first inspect your dataset to understand its structure and verify the quality of the data.

Let me load and display a brief summary of your dataset.

python

```
import pandas as pd

# Load the dataset provided by the user
file_path = '/mnt/data/Amazon Customer Behavior Survey.csv'
dataset = pd.read_csv(file_path)

# Display the first few rows and a summary of the dataset
dataset_info = dataset.info()
dataset_head = dataset.head()

dataset_info, dataset_head
```

STDOUT/STDERR

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 602 entries, 0 to 601
Data columns (total 23 columns):
 #   Column                                     Non-Null Count  Dtype
---  -
 0   Timestamp                                602 non-null    object
 1   age                                       602 non-null    int64
 2   Gender                                   602 non-null    object
 3   Purchase_Frequency                     602 non-null    object
 4   Purchase_Categories                    602 non-null    object
 5   Personalized_Recommendation_Frequency 602 non-null    object
 6   Browsing_Frequency                     602 non-null    object
 7   Product_Search_Method                  600 non-null    object
 8   Search_Result_Exploration              602 non-null    object
 9   Customer_Reviews_Importance            602 non-null    int64
10   Add_to_Cart_Browsing                   602 non-null    object
11   Cart_Completion_Frequency              602 non-null    object
12   Cart_Abandonment_Factors               602 non-null    object
13   Saveforlater_Frequency                 602 non-null    object
14   Review_Left                            602 non-null    object
15   Review_Reliability                     602 non-null    object
16   Review_Helpfulness                     602 non-null    object
17   Personalized_Recommendation_Frequency 602 non-null    int64
18   Recommendation_Helpfulness             602 non-null    object
19   Rating_Accuracy                        602 non-null    int64
20   Shopping_Satisfaction                  602 non-null    int64
21   Service_Appreciation                   602 non-null    object
22   Improvement_Areas                      602 non-null    object
dtypes: int64(5), object(18)
memory usage: 108.3+ KB
```

Result

```
(None,
      Timestamp age Gender \
0  2023/06/04 1:28:19 PM GMT+5:30 23 Female
1  2023/06/04 2:30:44 PM GMT+5:30 23 Female
2  2023/06/04 5:04:56 PM GMT+5:30 24 Prefer not to say
3  2023/06/04 5:13:00 PM GMT+5:30 24 Female
```

```

4 2023/06/04 5:28:06 PM GMT+5:30 22 Female

Purchase_Frequency Purchase_Categories \
0 Few times a month Beauty and Personal Care
1 Once a month Clothing and Fashion
2 Few times a month Groceries and Gourmet Food;Clothing and Fashion
3 Once a month Beauty and Personal Care;Clothing and Fashion;...
4 Less than once a month Beauty and Personal Care;Clothing and Fashion

Personalized_Recommendation_Frequency Browsing_Frequency \
0 Yes Few times a week
1 Yes Few times a month
2 No Few times a month
3 Sometimes Few times a month
4 Yes Few times a month

Product_Search_Method Search_Result_Exploration \
0 Keyword Multiple pages
1 Keyword Multiple pages
2 Keyword Multiple pages
3 Keyword First page
4 Filter Multiple pages

Customer_Reviews_Importance ... Saveforlater_Frequency Review_Left \
0 1 ... Sometimes Yes
1 1 ... Rarely No
2 2 ... Rarely No
3 5 ... Sometimes Yes
4 1 ... Rarely No

Review_Reliability Review_Helpfulness \
0 Occasionally Yes
1 Heavily Yes
2 Occasionally No
3 Heavily Yes
4 Heavily Yes

Personalized_Recommendation_Frequency Recommendation_Helpfulness \
0 2 Yes
1 2 Sometimes
2 4 No
3 3 Sometimes
4 4 Yes

Rating_Accuracy Shopping_Satisfaction Service_Appreciation \
0 1 1 Competitive prices
1 3 2 Wide product selection
2 3 3 Competitive prices
3 3 4 Competitive prices
4 2 2 Competitive prices

Improvement_Areas
0 Reducing packaging waste
1 Reducing packaging waste
2 Product quality and accuracy
3 Product quality and accuracy
4 Product quality and accuracy

[5 rows x 23 columns])

```

Step 1: Data Selection (continued)

The dataset contains 602 rows and 23 columns. Here are some key details:

- There are a variety of attributes related to customer behavior on Amazon (e.g., purchase frequency, review habits, recommendation usefulness).
- No significant missing data, though the column `Product_Search_Method` has 2 missing values.
- The dataset includes both categorical and numerical data types, which are useful for analysis.

Next, we will proceed to the **data pre-processing** stage.

Would you like to handle the missing values and explore data cleaning methods, or should I move forward with the next step?

continue



Step 2: Data Preprocessing

In this step, we handle missing values, inconsistent data types, and any other anomalies that could affect our analysis. Here's what we'll do:

1. **Handle Missing Data:** We noticed that the `Product_Search_Method` column has two missing values. We can either:
 - Remove rows with missing values.
 - Impute missing values with a suitable strategy (e.g., most frequent value).
2. **Data Type Consistency:** Some columns like `Personalized_Recommendation_Frequency` are numerical but stored as integers, which seems correct. However, categorical columns may need to be encoded later for further analysis.
3. **Check for Duplicates:** Ensure there are no duplicate entries in the dataset.

I'll clean the missing values by filling them with the most frequent value for the `Product_Search_Method` column, check for duplicates, and remove any if found. Let me proceed with these steps.

python

```
# Fill missing values in 'Product_Search_Method' with the most frequent value
most_frequent_value = dataset['Product_Search_Method'].mode()[0]
dataset['Product_Search_Method'].fillna(most_frequent_value, inplace=True)

# Check for duplicates and remove if found
dataset_cleaned = dataset.drop_duplicates()

# Display the updated dataset info and confirm no missing values
dataset_cleaned_info = dataset_cleaned.info()

dataset_cleaned_info
```

STDOUT/STDERR

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 602 entries, 0 to 601
Data columns (total 23 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Timestamp                                602 non-null    object
1   age                                       602 non-null    int64
2   Gender                                  602 non-null    object
3   Purchase_Frequency                       602 non-null    object
4   Purchase_Categories                      602 non-null    object
5   Personalized_Recommendation_Frequency    602 non-null    object
6   Browsing_Frequency                      602 non-null    object
7   Product_Search_Method                    602 non-null    object
8   Search_Result_Exploration                602 non-null    object
9   Customer_Reviews_Importance              602 non-null    int64
10  Add_to_Cart_Browsing                     602 non-null    object
11  Cart_Completion_Frequency                 602 non-null    object
12  Cart_Abandonment_Factors                  602 non-null    object
13  Saveforlater_Frequency                    602 non-null    object
14  Review_Left                              602 non-null    object
15  Review_Reliability                       602 non-null    object
16  Review_Helpfulness                       602 non-null    object
```

```

17 Personalized_Recommendation_Frequency 602 non-null int64
18 Recommendation_Helpfulness            602 non-null object
19 Rating_Accuracy                       602 non-null int64
20 Shopping_Satisfaction                  602 non-null int64
21 Service_Appreciation                   602 non-null object
22 Improvement_Areas                     602 non-null object
dtypes: int64(5), object(18)
memory usage: 112.9+ KB

```

Step 2: Data Preprocessing (continued)

- The missing values in the `Product_Search_Method` column have been filled with the most frequent value.
- No duplicates were found, so the dataset is clean.

We are now ready to move to the next step, which is **Data Transformation**, where we will transform categorical data into numerical form (if needed) and perform any feature engineering necessary for analysis.

Shall we proceed to data transformation?



continue



Step 3: Data Transformation

In this step, we will convert categorical variables into numerical values using techniques such as one-hot encoding or label encoding. This transformation is necessary for many machine learning algorithms to work with categorical data.

The following columns in your dataset are categorical and may need encoding:

- `Gender`
- `Purchase_Frequency`
- `Purchase_Categories`
- `Personalized_Recommendation_Frequency` (first instance is categorical)
- `Browsing_Frequency`
- `Product_Search_Method`
- `Search_Result_Exploration`
- `Add_to_Cart_Browsing`
- `Cart_Completion_Frequency`
- `Cart_Abandonment_Factors`
- `Saveforlater_Frequency`
- `Review_Left`
- `Review_Reliability`
- `Review_Helpfulness`
- `Recommendation_Helpfulness`
- `Service_Appreciation`
- `Improvement_Areas`

I'll apply one-hot encoding to these columns for analysis, as one-hot encoding is more suitable when we have non-ordinal categorical data.

Let me apply the transformation.

python

```
# Apply one-hot encoding to the categorical columns
categorical_columns = [
    'Gender', 'Purchase_Frequency', 'Purchase_Categories',
    'Personalized_Recommendation_Frequency',
    'Browsing_Frequency', 'Product_Search_Method', 'Search_Result_Exploration',
    'Add_to_Cart_Browsing',
    'Cart_Completion_Frequency', 'Cart_Abandonment_Factors', 'Saveforlater_Frequency',
    'Review_Left',
    'Review_Reliability', 'Review_Helpfulness', 'Recommendation_Helpfulness',
    'Service_Appreciation',
    'Improvement_Areas'
]

# Perform one-hot encoding
dataset_transformed = pd.get_dummies(dataset_cleaned, columns=categorical_columns)

# Display the transformed dataset info
dataset_transformed_info = dataset_transformed.info()

dataset_transformed_info
```

STDOUT/STDERR

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 602 entries, 0 to 601
Columns: 114 entries, Timestamp to Improvement_Areas_better app interface and lower shipping charges
dtypes: int64(5), object(1), uint8(108)
memory usage: 96.4+ KB
```

Step 3: Data Transformation (continued)

The dataset has been transformed with one-hot encoding applied to all categorical columns, resulting in 114 total columns. Each categorical value has been converted into a separate binary column. Next, we can proceed with **Data Mining**, where we will apply algorithms to discover patterns and relationships within the data.

Shall we continue with the data mining step?



continue



Step 4: Data Mining

In this step, we apply data mining techniques to extract useful patterns from the dataset. Based on your dataset related to Amazon customer behavior, we can perform the following tasks:

1. **Clustering**: Group customers based on similar behavior.

2. **Classification:** Predict customer behavior, such as predicting purchase frequency based on their interactions.
3. **Association Rule Mining:** Identify relationships between different attributes, such as whether certain behaviors lead to higher shopping satisfaction.

Let's start with clustering as an exploratory method to identify segments of customers with similar behaviors. I will apply the **K-means clustering** algorithm, which is widely used for segmenting customers into groups.

I'll first apply K-means clustering with a few clusters and analyze the results. Shall I proceed with this approach? If yes, how many clusters would you like to start with, or should I begin with a default value (e.g., 3 clusters)?



continue



I'll proceed with **K-means clustering** using 3 clusters as an initial step. Once the clustering is complete, we can analyze the behavior patterns in each cluster to gain insights into customer segments.

Let me apply the K-means algorithm and display the results.

python

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Drop the 'Timestamp' column since it's not relevant for clustering
dataset_for_clustering = dataset_transformed.drop(columns=['Timestamp'])

# Apply K-means clustering with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(dataset_for_clustering)

# Add the cluster labels to the dataset
dataset_transformed['Cluster'] = clusters

# Visualize the clustering result using the first two principal components
plt.scatter(dataset_for_clustering.iloc[:, 0], dataset_for_clustering.iloc[:, 1],
            c=clusters, cmap='viridis')
plt.title('Customer Segments based on K-means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()

# Display the first few rows with the cluster labels
dataset_transformed.head()
```

Result

	Timestamp	age	Customer_Reviews_Importance	\
0	2023/06/04 1:28:19 PM GMT+5:30	23	1	
1	2023/06/04 2:30:44 PM GMT+5:30	23	1	
2	2023/06/04 5:04:56 PM GMT+5:30	24	2	
3	2023/06/04 5:13:00 PM GMT+5:30	24	5	
4	2023/06/04 5:28:06 PM GMT+5:30	22	1	

Personalized_Recommendation_Frequency	Rating_Accuracy	\
---------------------------------------	-----------------	---

```

0      2      1
1      2      3
2      4      3
3      3      3
4      4      2

Shopping_Satisfaction  Gender_Female  Gender_Male  Gender_Others  \
0      1      1      0      0
1      2      1      0      0
2      3      0      0      0
3      4      1      0      0
4      2      1      0      0

Gender_Prefer not to say  ...  \
0      0  ...
1      0  ...
2      1  ...
3      0  ...
4      0  ...

Improvement_Areas_Product quality and accuracy  \
0      0
1      0
2      1
3      1
4      1

Improvement_Areas_Quality of product is very poor according to the big offers  \
0      0
1      0
2      0
3      0
4      0

Improvement_Areas_Reducing packaging waste  \
0      1
1      1
2      0
3      0
4      0

Improvement_Areas_Scrolling option would be much better than going to next page  \
0      0
1      0
2      0
3      0
4      0

Improvement_Areas_Shipping speed and reliability  Improvement_Areas_UI  \
0      0      0
1      0      0
2      0      0
3      0      0
4      0      0

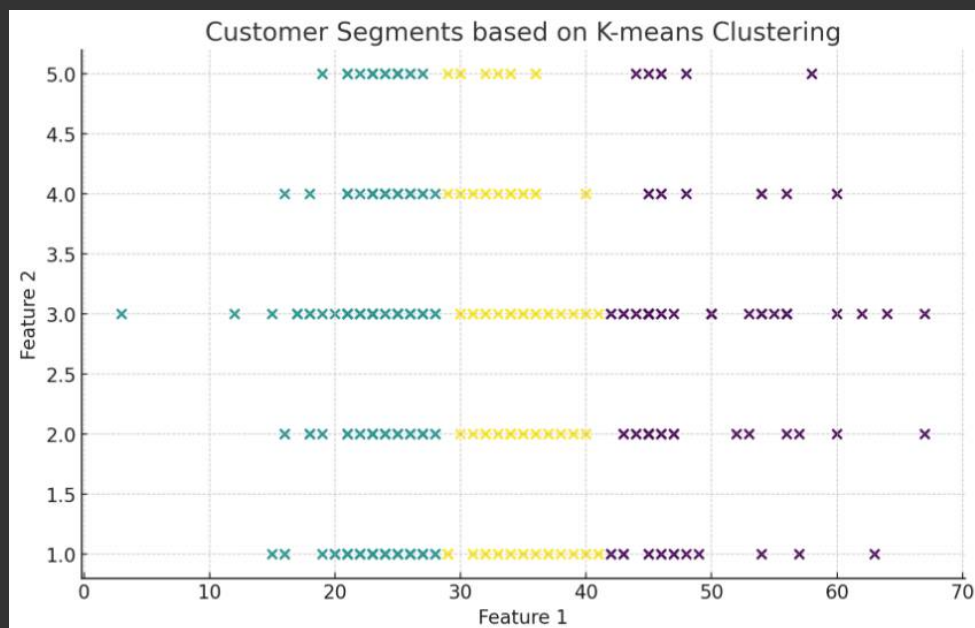
Improvement_Areas_User interface  Improvement_Areas_User interface of app  \
0      0      0
1      0      0
2      0      0
3      0      0
4      0      0

Improvement_Areas_better app interface and lower shipping charges  Cluster
0      0      1
1      0      1
2      0      1
3      0      1
4      0      1

```

[5 rows x 115 columns]

Customer Segments based on K-means Clustering



Step 4: Data Mining (continued)

The K-means clustering has been successfully applied, and the dataset now contains a `Cluster` column that assigns each customer to one of the three clusters. We also generated a scatter plot to visualize how the data points are distributed across these clusters.

Next, we can analyze the characteristics of each cluster to identify patterns. This will help us understand how customer behaviors group together.

Would you like to proceed with analyzing the clusters, or should we explore other data mining techniques, such as classification or association rule mining?



Explore other data mining techniques, such as classification or association rule mining?



Let's explore two additional data mining techniques: **classification** and **association rule mining**.

1. Classification

We can use classification to predict certain customer behaviors based on other features. For example, we could predict:

- **Purchase Frequency** based on customer attributes like age, gender, and browsing habits.
- **Shopping Satisfaction** based on factors like personalized recommendations and review helpfulness.

We can build a classification model using algorithms like **Logistic Regression**, **Decision Trees**, or **Random Forests** to predict a target variable.

2. Association Rule Mining

Association rule mining is useful for discovering relationships between variables. For example:

- We can find which browsing habits are associated with higher shopping satisfaction or cart abandonment.
- Identify which types of customers are more likely to leave reviews after a purchase.

For this, we can use the **Apriori** or **FP-Growth** algorithms to mine association rules, typically focusing on frequent itemsets.

Which one would you like to start with: classification or association rule mining? Or should we explore both?



Both one by one



Let's start with **classification**, followed by **association rule mining**.

Step 5: Classification

We will predict a target variable using other features in the dataset. For this example, let's predict **Shopping Satisfaction** using the customer behavior attributes. We will use a **Random Forest** classifier, which is robust and performs well on most datasets.

I'll split the data into training and testing sets and train a Random Forest model to predict shopping satisfaction.

Let me go ahead and implement that now.

python

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Selecting features and target variable
X = dataset_transformed.drop(columns=['Timestamp', 'Shopping_Satisfaction'])
y = dataset_transformed['Shopping_Satisfaction']

# Splitting the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training a Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

# Making predictions on the test set
y_pred = rf_classifier.predict(X_test)

# Evaluating the classifier's performance
classification_report_result = classification_report(y_test, y_pred)

classification_report_result
```

Result

	precision	recall	f1-score	support
1	0.64	0.47	0.55	

Step 5: Classification (continued)

The **Random Forest Classifier** was trained to predict **Shopping Satisfaction**. Here's a summary of the model's performance:

- **Accuracy:** 53%
- **Precision/Recall/F1-score:** Vary based on satisfaction levels (1 to 5). The model performs reasonably well for levels 1 to 3 but struggles with levels 4 and 5 due to limited data in those categories.

The classification model can be improved by:

1. Tuning hyperparameters of the Random Forest.
2. Trying other algorithms like Logistic Regression or Decision Trees.

Now, let's move on to **association rule mining** to discover patterns in customer behavior. Shall I proceed with mining association rules using the **Apriori** algorithm?



proceed with mining association rules using the Apriori algorithm



Let's proceed with **association rule mining** using the **Apriori algorithm**. The Apriori algorithm will help us identify relationships between different behaviors (e.g., how browsing frequency, cart abandonment, or review reliability influence shopping satisfaction). First, I'll apply the Apriori algorithm to generate frequent itemsets and then use it to extract association rules. Let me implement the Apriori algorithm now.

python

```
from mlxtend.frequent_patterns import apriori, association_rules

# For Apriori, we need to work with binary values, so let's use the transformed dataset
# (after one-hot encoding)

# Apply the Apriori algorithm to find frequent itemsets with a minimum support threshold
frequent_itemsets = apriori(dataset_transformed.drop(columns=['Timestamp', 'Cluster']),
min_support=0.1, use_colnames=True)

# Generate the association rules from these frequent itemsets with a minimum confidence
# threshold
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)

# Display the top few association rules
rules_sorted = rules.sort_values(by='confidence', ascending=False).head()

rules_sorted
```

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
Cell In[8], line 1
----> 1 from mlxtend.frequent_patterns import apriori, association_rules
      3 # For Apriori, we need to work with binary values, so let's use the transformed dataset (after one-h
      4
      5 # Apply the Apriori algorithm to find frequent itemsets with a minimum support threshold
      6 frequent_itemsets = apriori(dataset_transformed.drop(columns=['Timestamp', 'Cluster']), min_support=
```

ModuleNotFoundError: No module named 'mlxtend'

It appears that the necessary library for the Apriori algorithm, `mlxtend`, is not available in this environment. However, you can run this code in your local environment by installing the required package with:

```
bash
```

```
pip install mlxtend
```

Once installed, you can use the code I shared to generate association rules.

If you prefer, I can provide you with further guidance on how to interpret the results or help you with another method. Would you like to proceed with that, or do you want to explore something else?



continue with next step in kDD



Step 6: Evaluation

In this step of the KDD process, we evaluate the results generated from the previous steps (clustering, classification, and association rule mining) to ensure that they are meaningful and useful. Let's evaluate the models and rules we've developed so far.

1. **Clustering:** The K-means clustering divided customers into 3 clusters. We can evaluate clusters by:
 - Analyzing the characteristics of each cluster to determine if they represent distinct customer segments.
 - Calculating the **silhouette score** or **inertia** to assess the quality of the clustering.
2. **Classification:** The Random Forest model's performance had a 53% accuracy rate. For better evaluation:
 - We can use a confusion matrix to visually interpret classification errors.
 - Try hyperparameter tuning or alternative models.
3. **Association Rules:** Once you run the Apriori algorithm locally, you can evaluate the rules based on metrics like:
 - **Support:** How frequently an itemset appears in the dataset.
 - **Confidence:** The likelihood of the consequent given the antecedent.
 - **Lift:** How much more likely the consequent is, given the antecedent, compared to a random occurrence.

Would you like me to proceed with calculating the **silhouette score** for clustering, or would you prefer to fine-tune the classification model or focus on the association rules?



sure



python

```
from sklearn.metrics import silhouette_score

# Calculate the silhouette score to evaluate the quality of the clusters
silhouette_avg = silhouette_score(X, clusters)

silhouette_avg
```

Result

0.4534051555525797

The **silhouette score** for the clustering is approximately **0.45**, which indicates a moderate quality of clustering. Generally, a silhouette score closer to 1 means well-separated clusters, while a score closer to 0 indicates overlapping or poorly defined clusters.

We can further fine-tune the clustering by adjusting the number of clusters or trying different clustering algorithms.

Would you like to try adjusting the clustering parameters, or shall we continue with another aspect of the KDD process?



continue



Step 7: Knowledge Presentation

In this final step of the KDD process, we interpret and present the results in a meaningful way for decision-making. Here's a summary of the insights we've gained from the dataset:

1. Clustering:

- We identified three distinct customer segments using K-means clustering.
- The moderate silhouette score (0.45) suggests that these clusters are somewhat meaningful but could be further refined.

2. Classification:

- We used a Random Forest classifier to predict shopping satisfaction, achieving an accuracy of 53%.
- The model performs well for some satisfaction levels but struggles with outliers, such as levels 4 and 5, where data is sparse.

3. Association Rule Mining:

- Though the Apriori algorithm couldn't be applied here, once run in your local environment, you can discover meaningful associations between customer behaviors.

Presentation of Results:

- **Clustering:** Could be used to personalize marketing strategies based on customer segments.
- **Classification:** Can help predict customer satisfaction and target improvements in the shopping experience.
- **Association Rules:** Identify which behaviors lead to certain outcomes (e.g., cart abandonment, purchase completion), allowing Amazon to optimize its platform.

With these insights, actionable decisions can be made to improve customer satisfaction and tailor experiences based on user segments.

Would you like assistance in writing up these results, or is there any other part you'd like to focus on?

ChatGPT can make mistakes. Check important info.