Intellihack 5.0

Task – 2

Customer Segmentation

Team	:	Outlier Rejects
Author	:	Sanila Wijesekara
Date	:	10-03-2025

Table of Contents

1. Introduction

Customer segmentation is a crucial technique in e-commerce that helps businesses understand different customer behaviors and tailor marketing strategies accordingly. This project focuses on clustering customers based on their purchasing patterns, browsing behavior, and discount usage. Using machine learning techniques, we aim to identify distinct customer segments such as Bargain Hunters, High Spenders, and Window Shoppers. The insights derived from this analysis can help businesses optimize their pricing strategies, personalize marketing campaigns and enhance customer experiences.

2. Dataset Overview

The dataset contains 6 features:

- customer_id : Unique id for the customer.
- total_purchases : Total number of purchases made by the customer.
- avg_cart_value : Average value of items in the customer's cart.
- total_time_spent : Total time spent on the platform (in minutes).
- product_click : Number of products viewed by the customer.
- discount_count : Number of times the customer used a discount code.

```
df.head()
```

	total_purchases	avg_cart_value	total_time_spent	product_click	discount_counts	customer_id
0	7.0	129.34	52.17	18.0	0.0	CM00000
1	22.0	24.18	9.19	15.0	7.0	CM00001
2	2.0	32.18	90.69	50.0	2.0	CM00002
3	25.0	26.85	11.22	16.0	10.0	CM00003
4	7.0	125.45	34.19	30.0	3.0	CM00004

3. Data Preprocessing

3.1 Handling Missing Values

- Missing data detected in columns:
  - total_purchases
  - avg_cart_value
  - product_click

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   total_purchases   979 non-null    float64
 1   avg_cart_value    979 non-null    float64
 2   total_time_spent  999 non-null    float64
 3   product_click     979 non-null    float64
 4   discount_counts   999 non-null    float64
 5   customer_id       999 non-null    object
dtypes: float64(5), object(1)
memory usage: 47.0+ KB
```

- Imputation Strategy: Mean imputation

Data Cleaning

```
[ ] def clean(data):
      data['total_purchases'].fillna(data['total_purchases'].mean(), inplace = True)
      data['avg_cart_value'].fillna(data['avg_cart_value'].mean(), inplace = True)
      data['product_click'].fillna(data['product_click'].mean(), inplace = True)
```

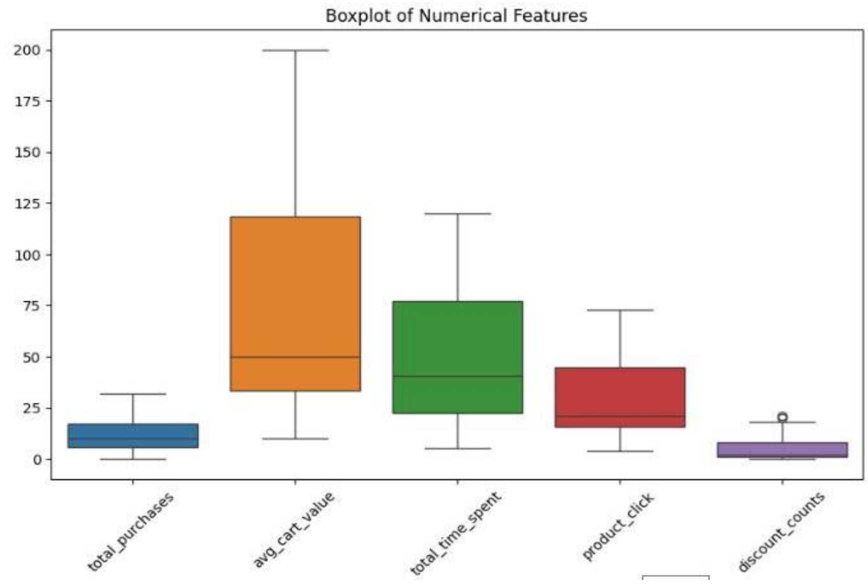
```
[ ] clean(df)
```

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   total_purchases   999 non-null    float64
 1   avg_cart_value    999 non-null    float64
 2   total_time_spent  999 non-null    float64
 3   product_click     999 non-null    float64
 4   discount_counts   999 non-null    float64
 5   customer_id       999 non-null    object
dtypes: float64(5), object(1)
memory usage: 47.0+ KB
```

3.2 Finding Incorrect Data Entries

- Identified outliers using boxplots and IQR method.



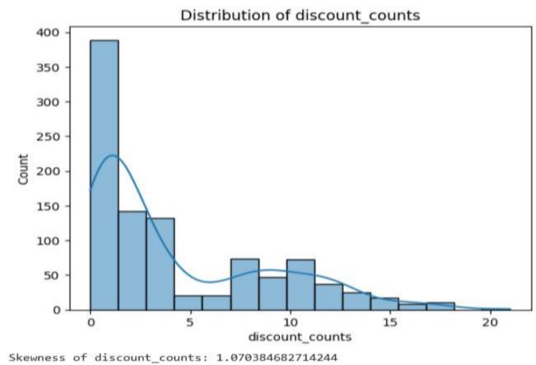

```python
def detect_outliers_iqr(df, feature):
    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[feature] < lower_bound) | (df[feature] > upper_bound)]
    return outliers

for feature in features:
    outliers = detect_outliers_iqr(df, feature)
    print(f"{feature}: {len(outliers)} outliers detected")
```

```
total_purchases: 0 outliers detected
avg_cart_value: 0 outliers detected
total_time_spent: 0 outliers detected
product_click: 0 outliers detected
discount_counts: 2 outliers detected
```

- These outliers in discount_counts can be the customers that legitimately use discounts heavily. So keeping them is the best choice.

3.3 Data transformation and Scaling

- Applied a logarithmic transformation to the **discount_counts** feature because the feature is highly skewed.



Before transformation

- Feature Scaling: Applied Standard Scaler for normalization.

Data Preprocessing

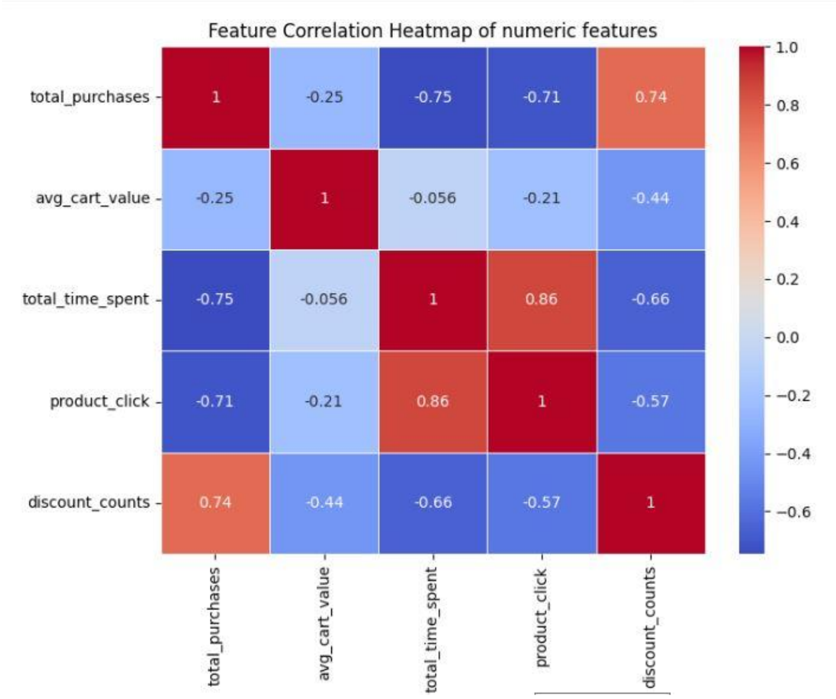
```
[ ] from sklearn.preprocessing import StandardScaler
```

```
[ ] def preprocess(data):
      # Transform the discount_counts column
      data['discount_counts'] = np.log1p(data['discount_counts'])  # log(1 + x) to handle zeros

      # Droped non-numeric columns
      data = data.drop(['customer_id','discount_category','purchase_category',
                        'spending_category','engagement_category','browsing_category'], axis=1)


      # Feature scaling
      features = ["total_purchases", "avg_cart_value", "total_time_spent", "product_click", "discount_counts"]
      scaler = StandardScaler()
      data[features] = scaler.fit_transform(data[features])

      return data
```

```
[ ] df_scaled = preprocess(df)
```

4. Exploratory Data Analysis (EDA)

4.1. Correlation Matrix

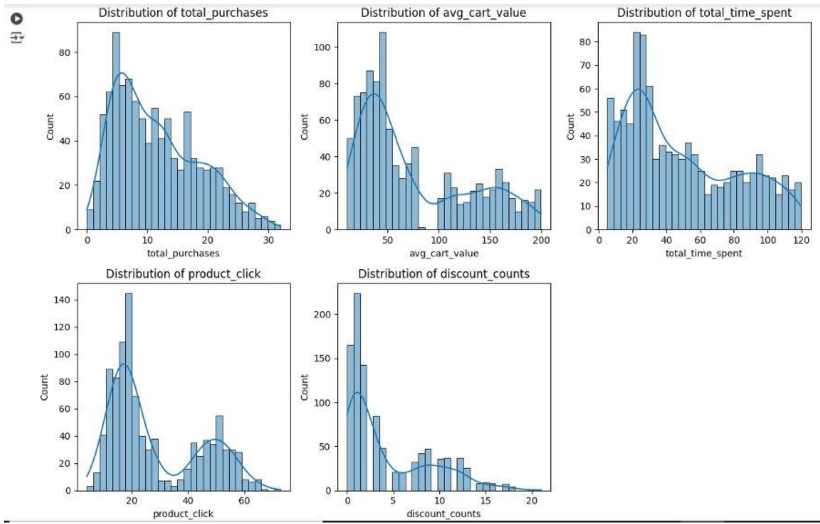


❖ Key Observations:

- product_click shows the strongest correlation with total_time_spent.

4.2. Data Visualization

Box plots



Key Observations:

- The majority of customers purchase a small number of items (0–10).
- Most customers prefer cheaper items priced between $25–$50.
- Some customers purchase expensive items, but items valued near $100 are rarely bought.
- Most customers spend around 20 minutes on the platform, though some spend 100–120 minutes.
- The majority view 10–20 items, but a significant number also view 50 items.
- Customers rarely view a moderate number of items (30–40).
- Most customers use only a few discount codes

4.3 Feature Patterns

Discount usage patterns

```
[ ] df["discount_category"] = pd.cut(df["discount_counts"], bins=[-1, 1, 5, df["discount_counts"].max()],
                                  labels=["Low", "Moderate", "High"])

    # Analyze discount usage patterns
    df.groupby("discount_category")['customer_id'].count()
```

discount_category	customer_id
Low	389
Moderate	295
High	315

dtype: int64

- The majority of customers have applied discount codes to only a few times.

Purchasing patterns

```
[ ] df["purchase_category"] = pd.qcut(df["total_purchases"], q=3, labels=["Low", "Medium", "High"])

    # Check average cart value and discount usage across purchase groups
    df.groupby("purchase_category")[["avg_cart_value", "discount_counts"]].mean()
```

purchase_category	avg_cart_value	discount_counts
Low	67.234578	1.209809
Medium	118.002860	3.089231
High	40.249186	9.319218

- Customers who purchase a large number of items tend to buy cheaper products.
- Customers who purchase a small number of items prefer medium-value products.
- Customers who purchase a moderate number of items tend to buy high-value products.

Spending patterns

```
[ ] df["spending_category"] = pd.qcut(df["avg_cart_value"], q=3, labels=["Low", "Medium", "High"])

    # Compare purchase behavior across spending levels
    df.groupby("spending_category")[["total_purchases", "discount_counts", "total_time_spent"]].mean()
```

spending_category	total_purchases	discount_counts	total_time_spent
Low	15.180180	7.195195	39.989820
Medium	9.487747	3.792793	66.576036
High	10.045045	1.951952	41.480420

- Customers who spend less money tend to use discount codes frequently.
- Customers who spend more money rarely use discount codes.

Engagement patterns

```
[ ] df["engagement_category"] = pd.qcut(df["total_time_spent"], q=3, labels=["Low", "Medium", "High"])

    # Analyze how engagement relates to spending
    df.groupby("engagement_category")[["total_purchases", "avg_cart_value", "product_click"]].mean()
```

engagement_category	total_purchases	avg_cart_value	product_click
Low	18.404257	46.797188	15.941576
Medium	11.302534	130.653721	19.399220
High	4.963528	49.426666	49.393964

Browsing patterns

```
df["browsing_category"] = pd.qcut(df["product_click"], q=3, labels=["Low", "Medium", "High"])

# Check if browsing affects purchases
df.groupby("browsing_category")[["total_purchases", "discount_counts"]].mean()
```

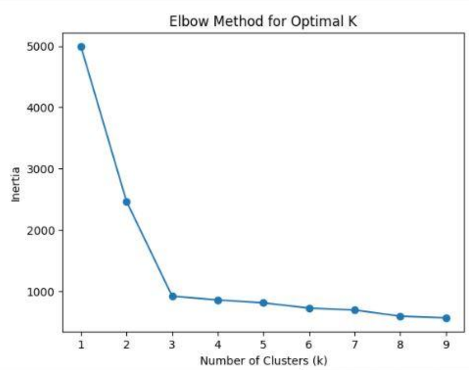
browsing_category	total_purchases	discount_counts
Low	17.168639	7.754438
Medium	12.521033	4.069909
High	4.930723	1.051205

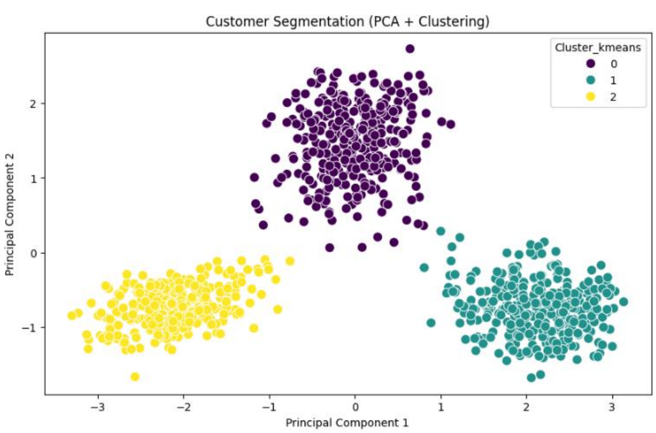
- Customers who spend more time on the platform tend to purchase fewer items.
- Customers who spend less time on the platform tend to purchase more items.

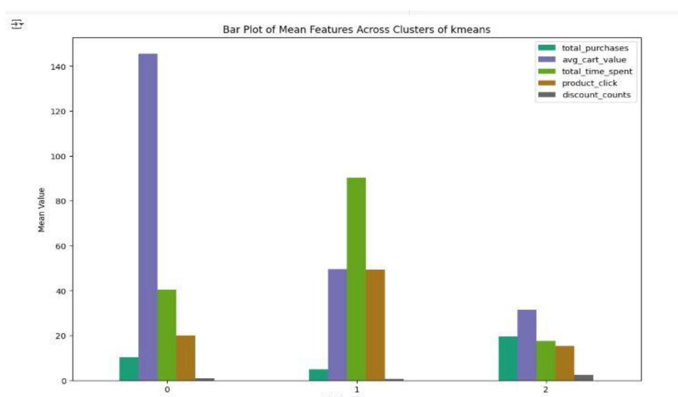
5. Clustering (K-Means)

K-Means clustering was applied to segment customers based on their behavior. After normalizing the data, we determined the optimal number of clusters using the Elbow Method. It suggested **k=3**. These clusters were visualized using Principal Component Analysis (PCA), showing clear separation between the groups.

```
inertia = []
K_range = range(1, 10)  # Test different cluster numbers
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)

plt.plot(K_range, inertia, marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.title("Elbow Method for Optimal K")
plt.show()
```





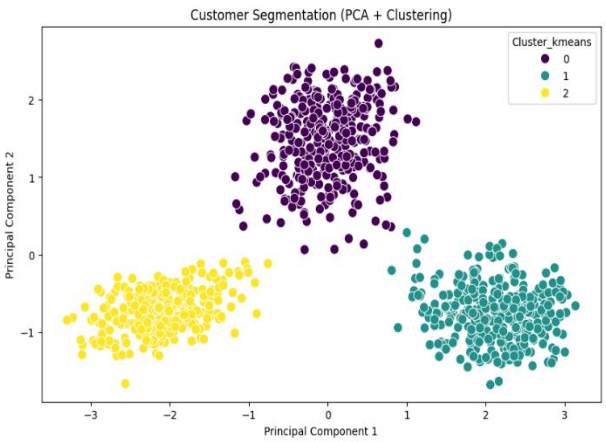


PCA visualization of K-Means clustering

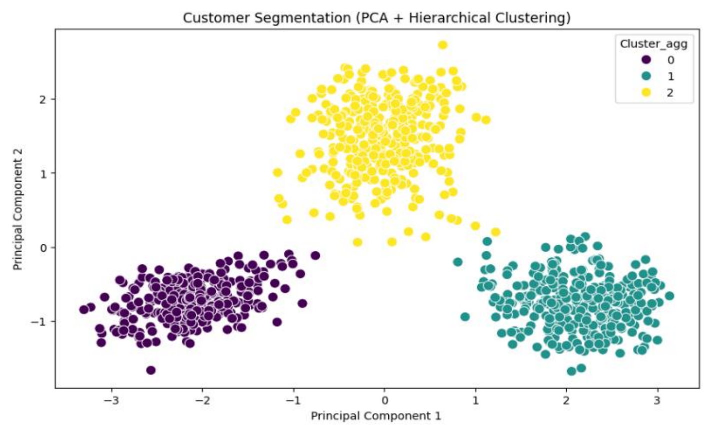
6. Model Selection

Given that we are dealing with an unsupervised learning task, where we need to identify hidden patterns without predefined labels, clustering algorithms were chosen. After comparing several algorithms, the following were considered:

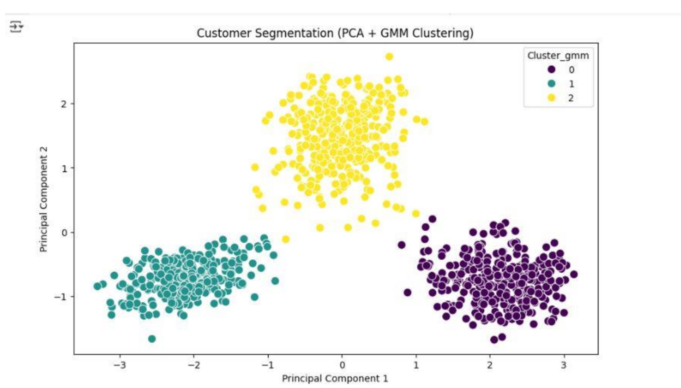
- **K-Means Clustering**: A popular clustering algorithm that is efficient and works well for spherical clusters.
- **Hierarchical(Agglomerative) Clustering**: Good for visualizing the clusters through a dendrogram.
- **Gaussian-Mixture model** : A probabilistic model that assumes data is generated from multiple Gaussian distributions.



PCA visualization of K-Means clustering



PCA visualization of Hierarchical Clustering



PCA visualization of Gaussian-Mixture model

7. Model Evaluation

After applying the K-Means clustering algorithm with three clusters, we evaluated the results by:

- **Silhouette Score** : Measures how similar a data point is to its own cluster compared to other clusters, ranging from -1 to 1. A high average silhouette score indicated that the clusters were well-separated and coherent.
- **Davies-Bouldin Score**: Evaluates cluster quality based on the ratio of intra-cluster dispersion to inter-cluster distance (lower is better).

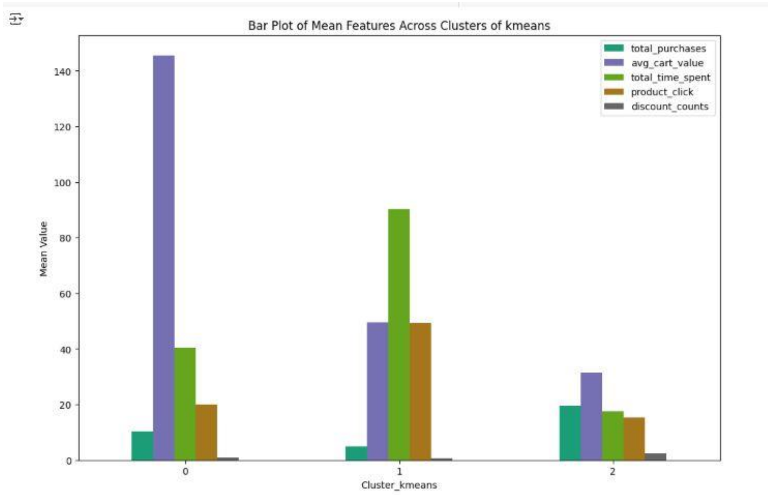
Clustering Algorithm	Silhouette Score	Davies-Bouldin Score
K-Means	0.6625	0.4639
Gaussian Mixture Model	0.6613	0.4650
Agglomerative Clustering	0.6605	0.4653

We decided to proceed with **K-Means Clustering**, as its Silhouette Score is higher and Davies-Bouldin Score is lower.

8. Cluster Interpretation

Based on the centroids and the characteristics of the clusters of K-Means, we identified the following segments:

1. **Bargain Hunters**: These customers tend to make frequent low-value purchases, rely heavily on discounts, and spend moderate time on the platform. They represent deal-seekers.
2. **High Spenders**: These customers make fewer but high-value purchases, spend moderate time on the platform, and rarely use discounts. They represent premium buyers.
3. **Window Shoppers**: These customers make few purchases but spend a lot of time browsing and viewing many products. They rarely use discounts. They represent potential leads who are not yet converted into buyers.



❖ **Key Observations:-**

- 0 belongs to **High Spenders**
- 1 belongs to **Window Shoppers**
- 2 belongs to **Bargain Hunters**

9. Conclusion

The customer segmentation task was successful in identifying three distinct groups:

1. **Bargain Hunters**
2. **High Spenders**
3. **Window Shoppers**

Each group exhibits unique behaviors, and this segmentation can be used to design targeted marketing strategies. For instance:

- **Bargain Hunters** could be targeted with special discounts or promotional offers.
- **High Spenders** could be offered premium products or exclusive offers.
- **Window Shoppers** could be engaged with personalized recommendations or reminders.

In future iterations, this model could be refined with additional features such as customer demographics or purchase frequency trends over time.

10. Future Work

- **Incorporate more features**: Demographic data could provide more granular insights into customer behavior.
- **Improve model performance**: Exploring more sophisticated clustering algorithms or using hybrid models to better handle different types of customer behavior.
- **Deployment**: Implementing the model into a recommendation system to automate personalized marketing.