INTELLIHACK 5.0

Task 02

Team Cognic AI

Introduction

Customer segmentation is an important strategy in e-commerce that helps businesses categorize customers based on their behaviors. Understanding these segments allows companies to provide better-targeted marketing, promotions, and personalized services.

This study analyzes customer data using K-Nearest Neighbors (KNN) clustering to identify three key customer segments:

- **Bargain Hunters**: Customers who often buy low-cost items and search for discounts.
- **High Spenders**: Customers who make fewer but more expensive purchases.
- **Window Shoppers**: Customers who browse a lot but rarely make purchases.

We'll explore the dataset, preprocess the data, apply clustering, evaluate the results, and visualize these customer segments.

We're given the additional information of behavior patterns of each cluster

Customer Type	Total Purchases	Avg. Cart Value	Time Spent	Product Click	Discount Usage
Bargain Hunters	High	Low	Moderate	Moderate	High
High Spenders	Moderate	High	Moderate	Moderate	Low
Window Shoppers	Low	Moderate	High	High	Low

Data Loading and Inspection

	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
...	...	...	...	...	...	...
994	5.0	64.64	72.70	50.0	1.0	CM00994
995	5.0	68.36	75.41	43.0	1.0	CM00995
996	18.0	19.53	28.77	18.0	8.0	CM00996
997	4.0	28.97	72.27	57.0	3.0	CM00997
998	29.0	39.29	9.99	16.0	11.0	CM00998

999 rows × 6 columns

The dataset consists of data on 999 customers, and 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.

	total_purchases	avg_cart_value	total_time_spent	product_click	discount_counts
count	979.000000	979.000000	999.000000	979.000000	999.000000
mean	11.570991	75.457978	49.348759	28.237998	4.313313
std	7.016327	55.067835	32.730973	16.296384	4.532772
min	0.000000	10.260000	5.120000	4.000000	0.000000
25%	6.000000	33.130000	22.375000	16.000000	1.000000
50%	10.000000	49.380000	40.360000	21.000000	2.000000
75%	17.000000	121.255000	77.170000	45.000000	8.000000
max	32.000000	199.770000	119.820000	73.000000	21.000000

Handling Missing Values

Missing data can negatively impact the model's performance. We identified missing values in the dataset and handled them accordingly.

```
[6]: # Check for missing values
     print(df.isnull().sum())

total_purchases     20
avg_cart_value      20
total_time_spent     0
product_click       20
discount_counts      0
customer_id          0
dtype: int64
```

```
[7]: for col in df.columns:
         if col != 'customer_id':
             missing_customers = df.loc[df[col].isnull(), 'customer_id']
             print(col)
             print([i[-3:] for i in missing_customers.to_list()])
             print("-" * 40)

total_purchases
['097', '139', '212', '253', '294', '310', '317', '353', '409', '425', '549', '555', '605', '622', '674', '765',
'920', '924', '936', '986']
----------------------------------------
avg_cart_value
['097', '139', '212', '253', '294', '310', '317', '353', '409', '425', '549', '555', '605', '622', '674', '765',
'920', '924', '936', '986']
----------------------------------------
total_time_spent
[]
----------------------------------------
product_click
['097', '139', '212', '253', '294', '310', '317', '353', '409', '425', '549', '555', '605', '622', '674', '765',
'920', '924', '936', '986']
----------------------------------------
discount_counts
[]
----------------------------------------
```

An initial search reveals that very few rows have missing values, and they are all the same rows.

Therefore, it was decided the best approach is to drop these rows entirely for model training.

```
df_drop = df.dropna()
```

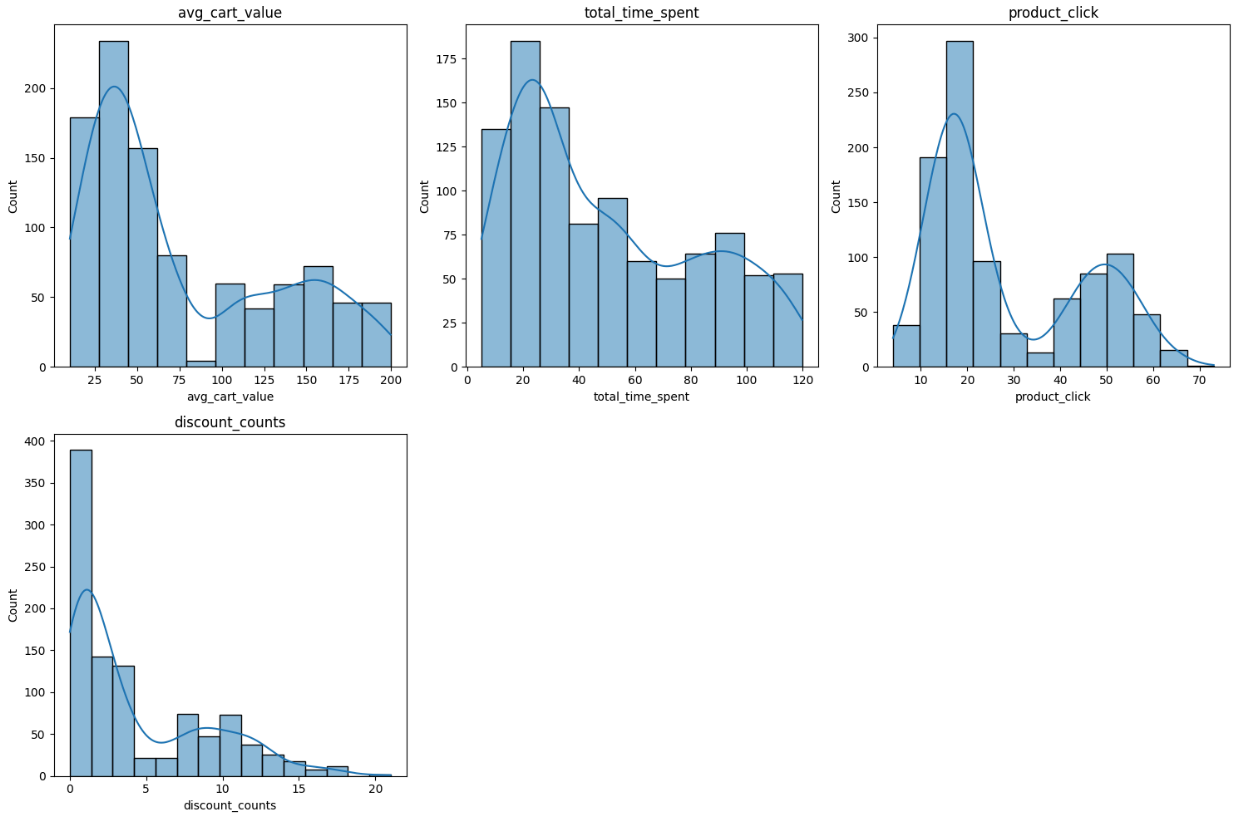
Since customer_id is an arbitrary identifier, we drop that column and proceed to EDA.

```
df_drop = df_drop.drop('customer_id', axis=1)
```

Exploratory Data Analysis

Data Distribution

Understanding the distribution of the features helps to identify patterns, trends, and potential issues.



Plotting the distribution of each feature reveals there are **no outliers** in the dataset. Therefore there is no need to handle outliers explicitly.

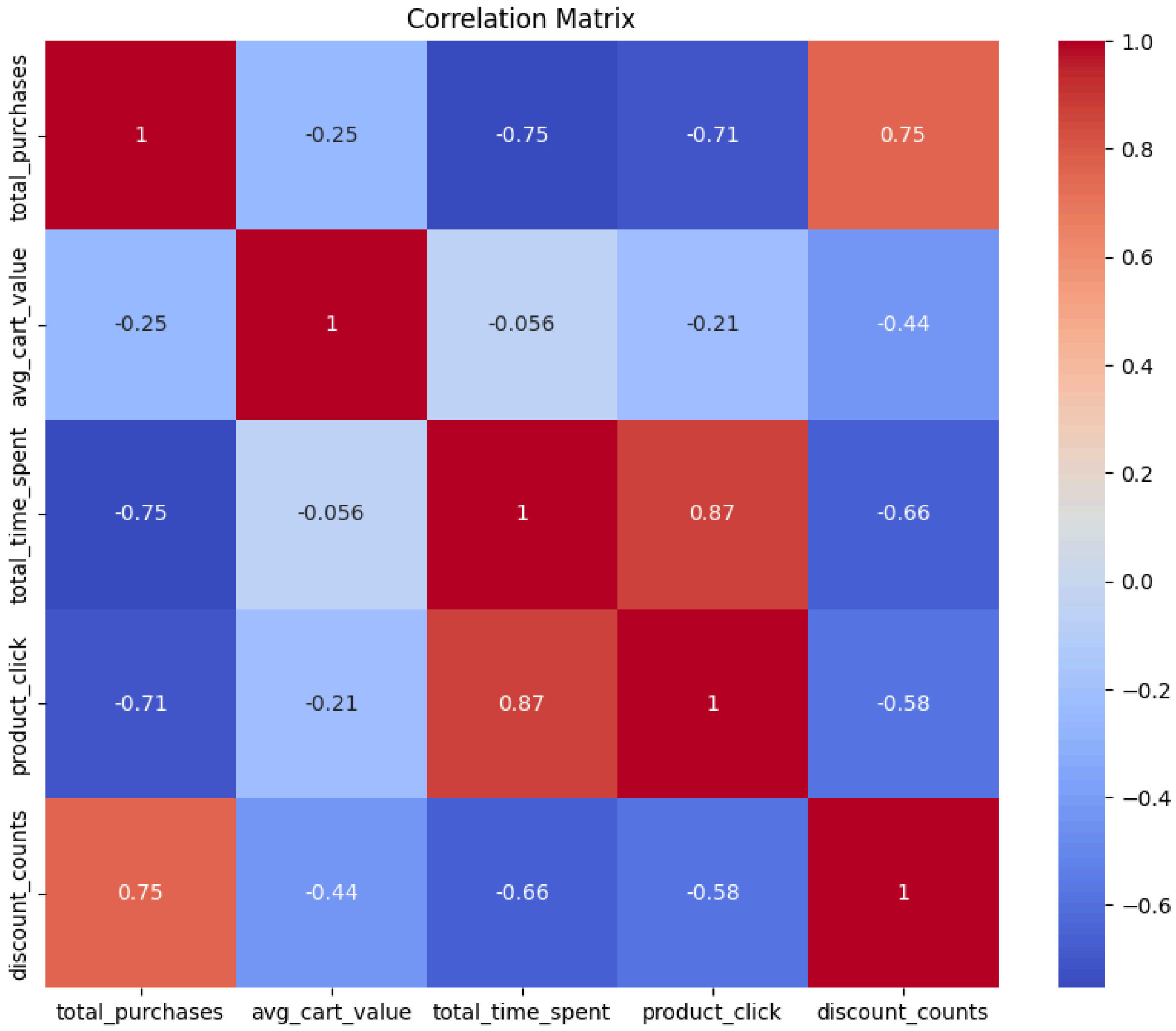
Since clustering methods like KNN rely on distance measurements, we standardized the features using **StandardScaler** from the **scikit-learn** library, which scales each feature to have a mean of 0 and a standard deviation of 1. This ensures that all features contribute equally to the distance calculation.

```
# Standardize the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_drop)
```

Data Correlation

We generated a correlation heatmap to visualize how strongly each feature relates to others. High correlation between **total purchases** and **discount_count** might suggest that customers who buy more frequently are also those who make the most use of discount codes. There is also a very high correlation between **total_time_spent** and **product_click**.

These results are consistent with our initial assumptions about the 3 clusters in the dataset.



Model Selection and Training

The **K-Nearest Neighbors** (KNN) algorithm is a distance-based clustering method that works by grouping data points based on their similarity to each other. KNN is ideal in this scenario.

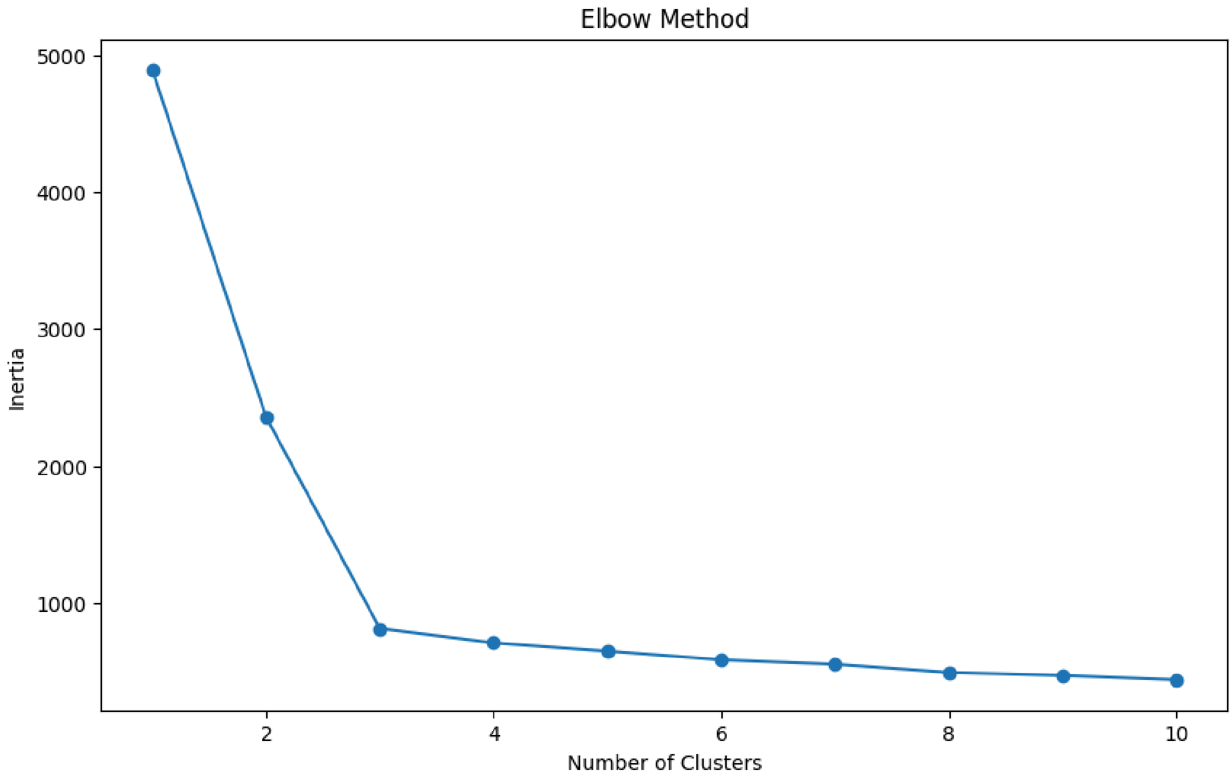
KNN is effective when the relationships in the data are non-linear, which is often the case with customer behavior.

Choosing the K Value

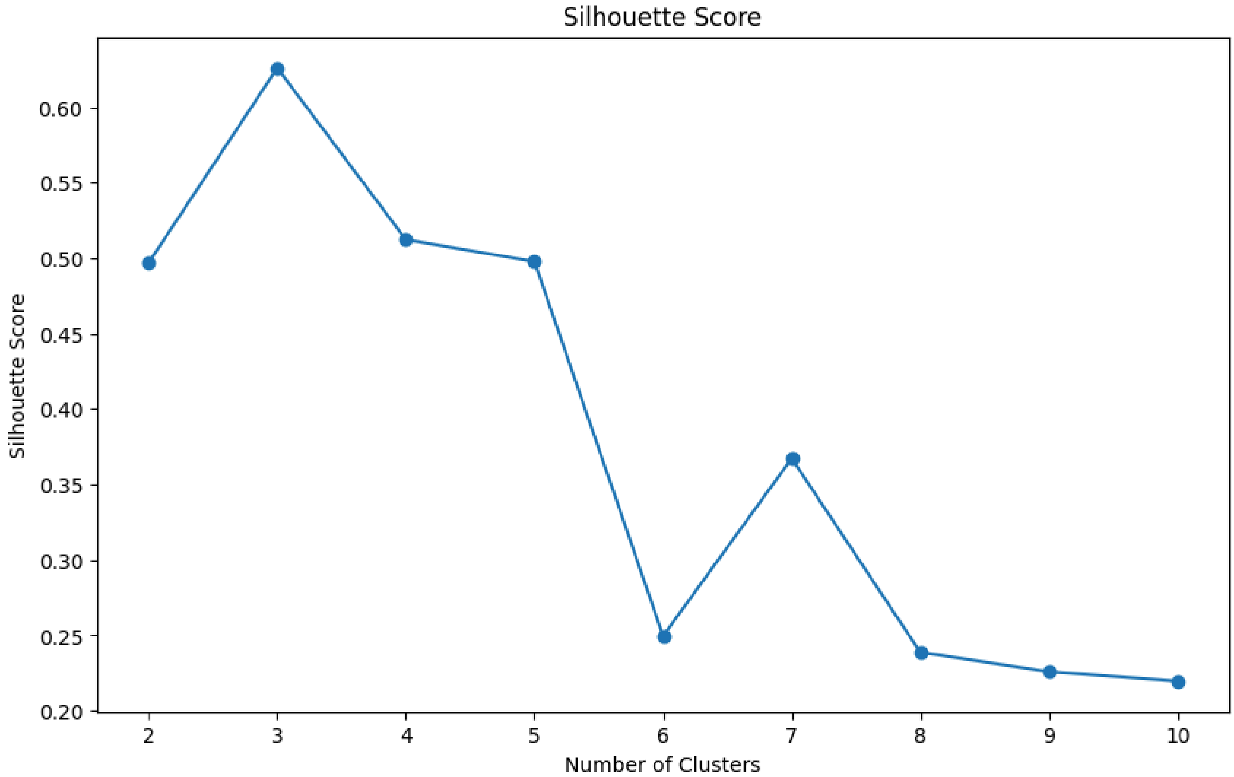
The task is to identify the 3 given clusters of customers. So naturally we set k = 3

Using the **elbow method** and **silhouette score**, we verify that this is indeed the optimal number of clusters.

Elbow method involves plotting the Within-Cluster Sum of Squares (WCSS) against the number of clusters. The point where the curve flattens forming an elbow indicates the optimal K.



Silhouette score measures how similar each point is to its own cluster compared to other clusters. A higher score indicates that the clusters are well-separated, with low overlap.



Model Training

After choosing the optimal K value, we implemented the KNN clustering model using the **scikit-learn** library. Since we already scaled the features, we apply the model directly.

```
# Train the KMeans model
kmeans = KMeans(n_clusters=3, random_state=1)
kmeans.fit(scaled_features)
```

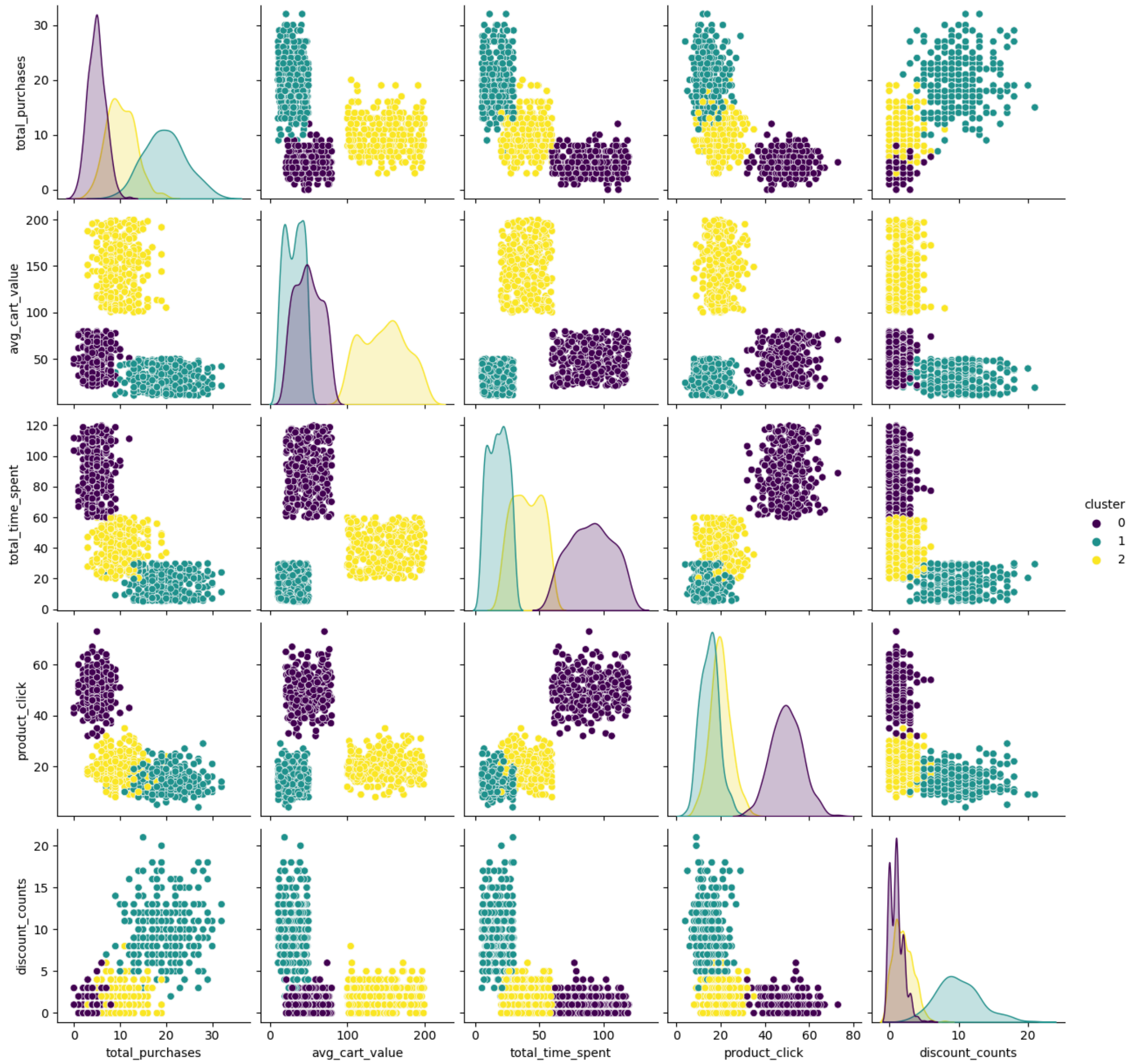
Here random_state is set explicitly for reproducibility.

Prediction and Validation

When we apply KNN clustering, the algorithm groups the data into clusters during the fitting process, so we do not need an additional predict step.

Visualizing the Clusters

To better understand the separation between the identified customer segments, we used a **pairplot** to visualize the relationships between key features, with the clusters color-coded. This plot allows us to identify the three customer groups - Bargain Hunters, High Spenders, and Window Shoppers.



In the pairplot, we can clearly observe that the three clusters follow the patterns initially given in the problem statement. Each customer segment forms distinct groupings, with their respective behaviors clearly separated across the different features. This visualization reinforces the validity of our clustering results, as the clusters align well with the characteristics given.

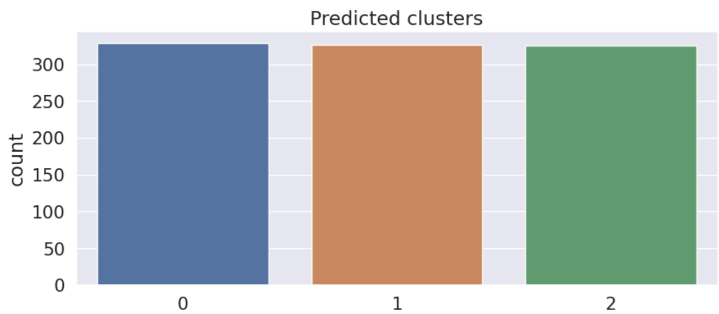
Customer Type	Total Purchases	Avg. Cart Value	Time Spent	Product Click	Discount Usage
Bargain Hunters	High	Low	Moderate	Moderate	High
High Spenders	Moderate	High	Moderate	Moderate	Low
Window Shoppers	Low	Moderate	High	High	Low

Analyzing the results of our clustering:

cluster	total_purchases	avg_cart_value	total_time_spent	product_click	discount_counts
0	4.862805	49.029848	90.114726	49.716463	1.030488
1	19.711656	30.399509	17.453988	14.944785	9.938650
2	10.175385	147.327169	40.284369	19.895385	1.972308

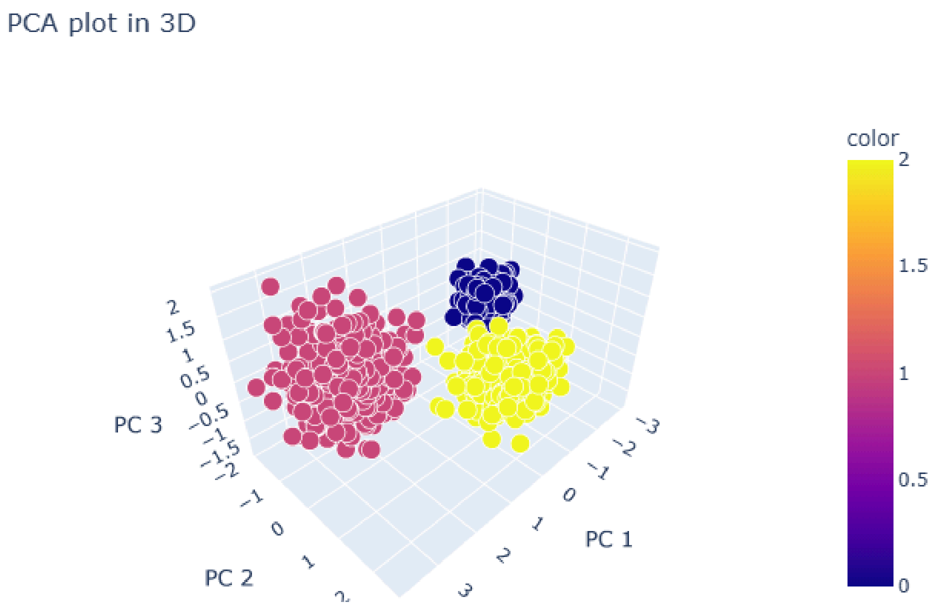
Hence we identify the clusters predicted by out model as

- 0 Purple - Window Shoppers
- 1 Cyan - Bargain Hunters
- 2 Yellow - High Spenders



Seems all the clusters are distributed equally.

Principle Component Analysis



The plot shows three well-separated groups of points.This confirms that our clustering algorithm has identified three clusters correctly in the dataset.

The clustering analysis was performed using **K-Nearest Neighbors (KNN)**, and did some additional testing using **Bayesian Gaussian Mixture Model (BGMM) and Gaussian Mixture Model (GMM)**. The results from all three methods were consistent, indicating that the underlying structure of the data is well-defined.