INTELLIHACK 5.0

INITIAL ROUND DELEGATES

Intellihack_DataDominators_TaskNumber2

Table of Contents

Section 01 – Introduction

1.1 What is Customer Segmentation

Customer segmentation is the practice of dividing a company's customers into different groups based on their purchasing behavior, browsing patterns, and other demographic data. These groups, or segments, allow businesses to target their marketing efforts more effectively, personalize their offerings, and optimize resource allocation.

1.2 Importance of Customer Segmentation

The key benefits of customer segmentation include

- Targeted Marketing - Tailor advertising and promotions to specific groups, increasing the likelihood of customer engagement.
- Customer Retention - Develop customized retention strategies for each segment, ensuring loyalty.
- Operational Efficiency - Focus resources on high-value customers, improving cost efficiency.
- Revenue Maximization - Personalize offers and improve conversion rates, boosting overall revenue.

1.3 Objectives of this Project

This project aims to perform customer segmentation on an e-commerce dataset. Using machine learning techniques such as K-Means clustering, we aim to identify three distinct customer segments:

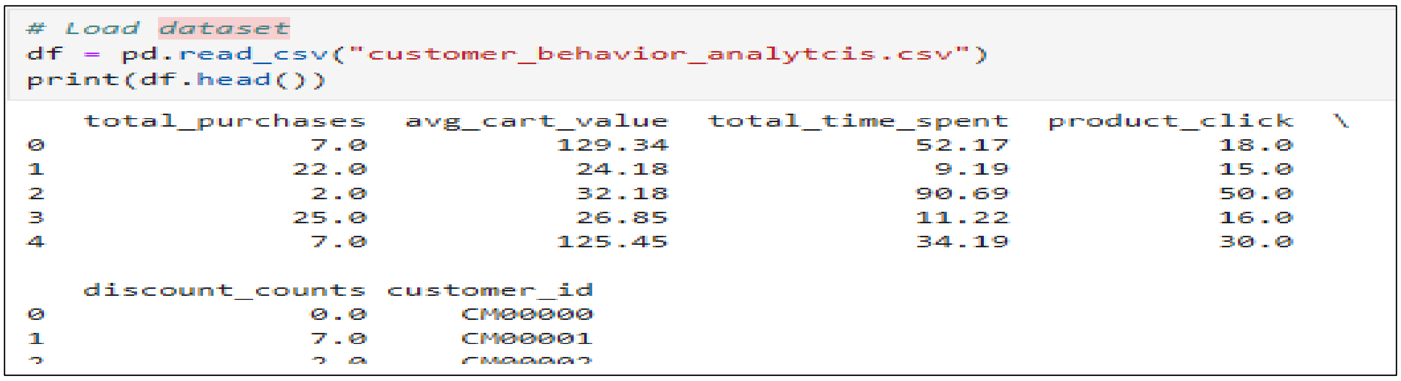
- Bargain Hunters - Customers who frequently purchase low-cost items and make extensive use of discounts.
- High Spenders - Customers who make fewer, but high-value, purchases.
- Window Shoppers - Customers who browse the platform often but rarely make purchases.

The goal is to use these segments to develop personalized strategies for improving customer engagement and boosting sales.

Section 02 – Exploratory Data Analysis (EDA)

2.1 Data Loading and Initial Exploration

After loading the dataset, we performed an initial exploration to understand its structure. The dataset consists of 999 rows and six features. The first few rows of the dataset look like

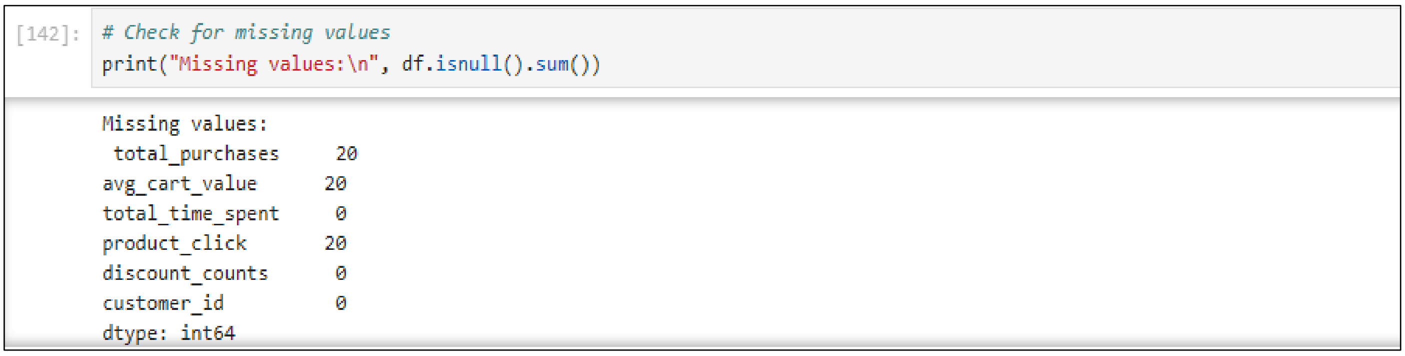


```
# Load dataset
df = pd.read_csv("customer_behavior_analytcis.csv")
print(df.head())
```

```
   total_purchases  avg_cart_value  total_time_spent  product_click  \
0              7.0          129.34             52.17           18.0   
1             22.0           24.18              9.19           15.0   
2              2.0           32.18             90.69           50.0   
3             25.0           26.85             11.22           16.0   
4              7.0          125.45             34.19           30.0   

   discount_counts customer_id  
0              0.0    CM00000  
1              7.0    CM00001  
2              2.0    CM00002  
```

2.2 Handling Missing Values

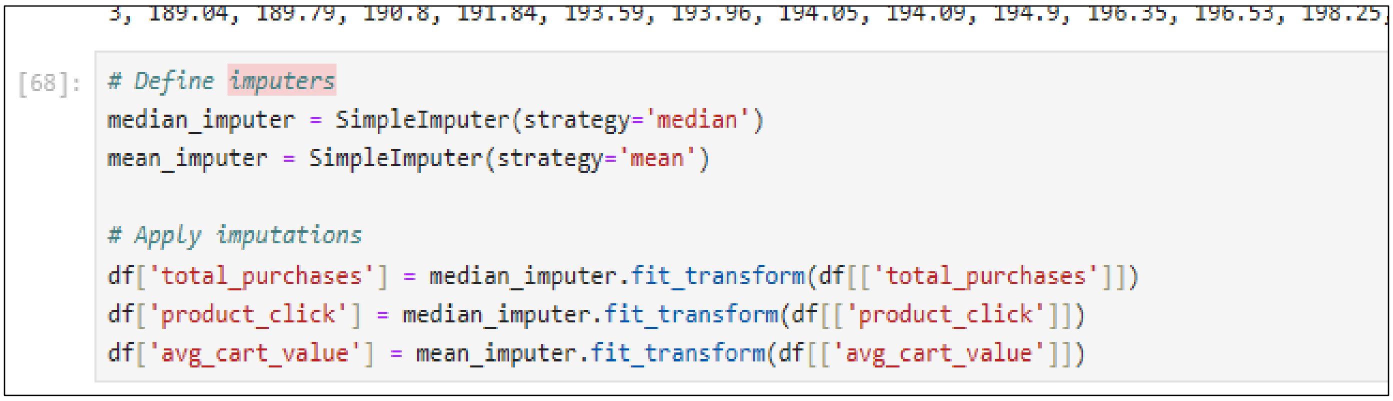


```
[142]: # Check for missing values
       print("Missing values:\n", df.isnull().sum())
```

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

We identified missing values in the total_purchases, avg_cart_value, and product_click columns. These were imputed using

- Median for total_purchases and product_click (due to skewed distribution).
- Mean for avg_cart_value.

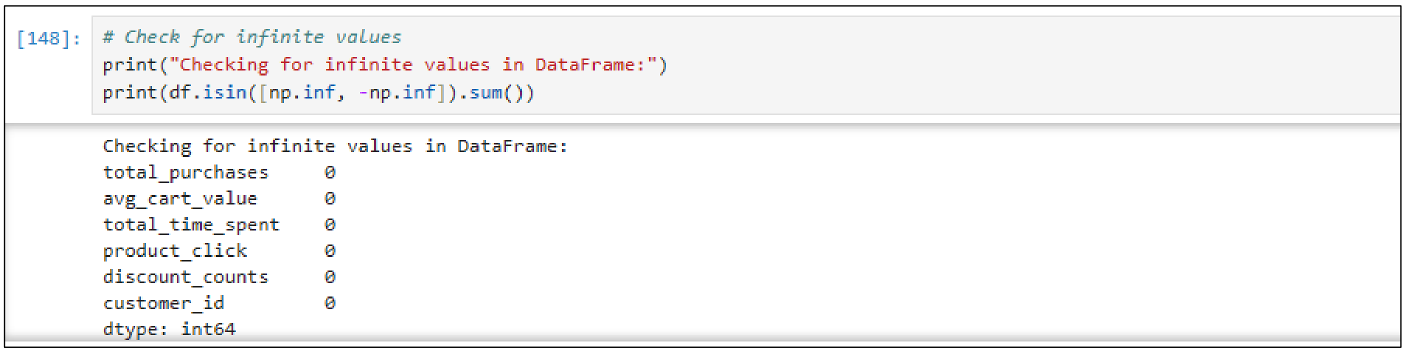


```
3, 189.04, 189.79, 190.8, 191.84, 193.59, 193.96, 194.05, 194.09, 194.9, 196.35, 196.53, 198.25,
```

```
[68]: # Define imputers
      median_imputer = SimpleImputer(strategy='median')
      mean_imputer = SimpleImputer(strategy='mean')

      # Apply imputations
      df['total_purchases'] = median_imputer.fit_transform(df[['total_purchases']])
      df['product_click'] = median_imputer.fit_transform(df[['product_click']])
      df['avg_cart_value'] = mean_imputer.fit_transform(df[['avg_cart_value']])
```

2.3 Checking for Infinite Values



```
[148]: # Check for infinite values
       print("Checking for infinite values in DataFrame:")
       print(df.isin([np.inf, -np.inf]).sum())
```

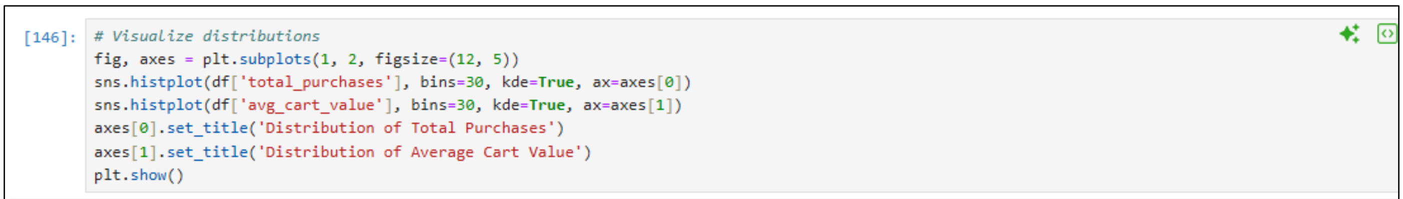
```
Checking for infinite values in DataFrame:
total_purchases     0
avg_cart_value      0
total_time_spent    0
product_click       0
discount_counts     0
customer_id         0
dtype: int64
```

A warning related to infinite values was dismissed after confirming there were no np.inf or -np.inf values present in the dataset.

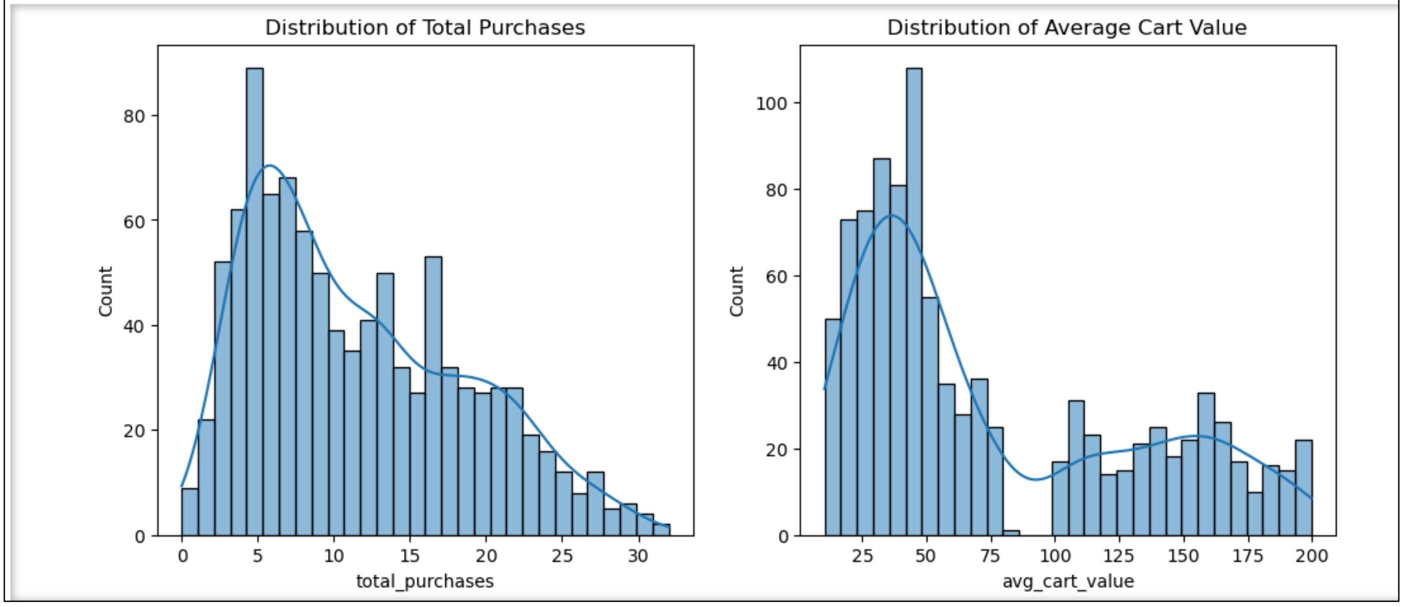
2.4 Data Visualizations

Visualizations helped identify distribution patterns and outliers in the data.

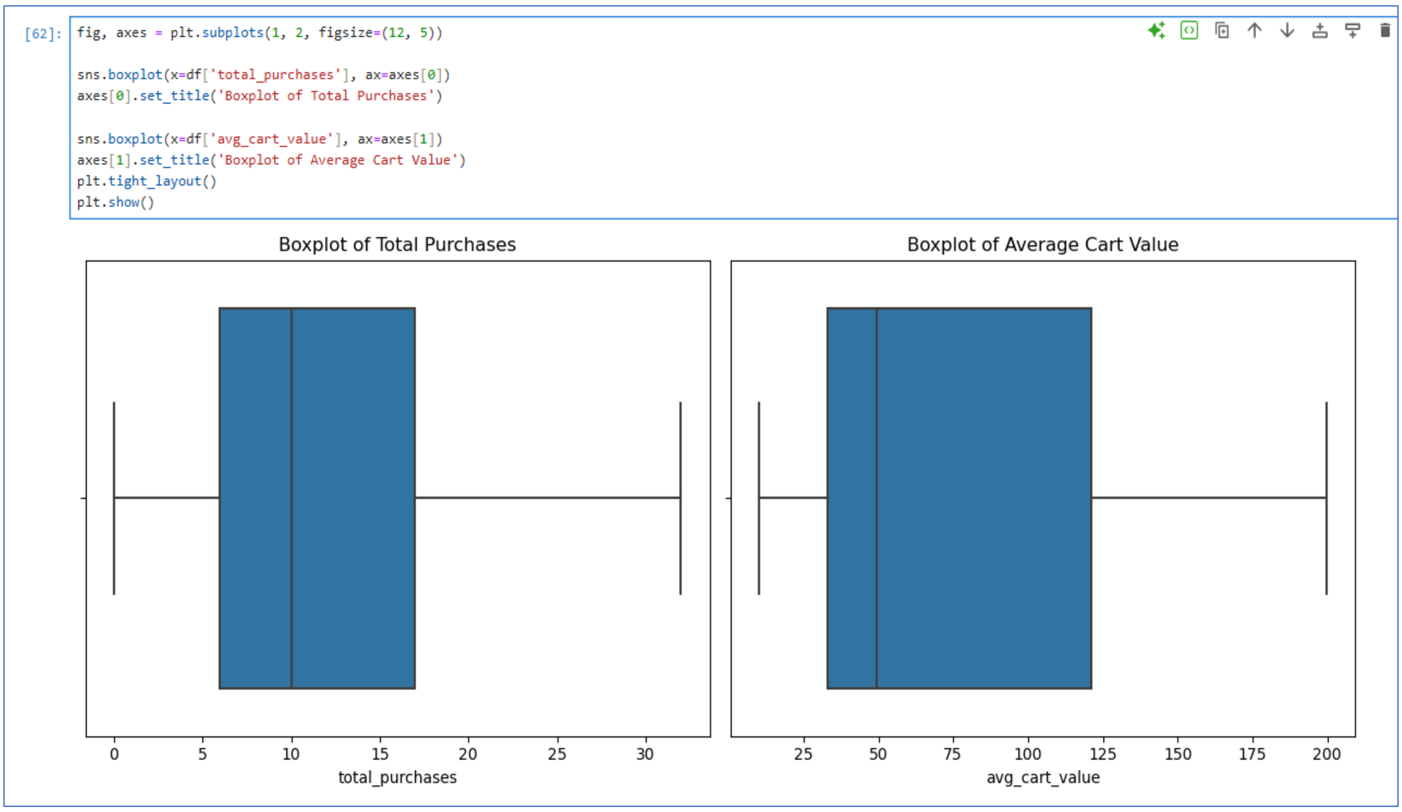
Histograms



```
[146]: # Visualize distributions
       fig, axes = plt.subplots(1, 2, figsize=(12, 5))
       sns.histplot(df['total_purchases'], bins=30, kde=True, ax=axes[0])
       sns.histplot(df['avg_cart_value'], bins=30, kde=True, ax=axes[1])
       axes[0].set_title('Distribution of Total Purchases')
       axes[1].set_title('Distribution of Average Cart Value')
       plt.show()
```



Boxplots

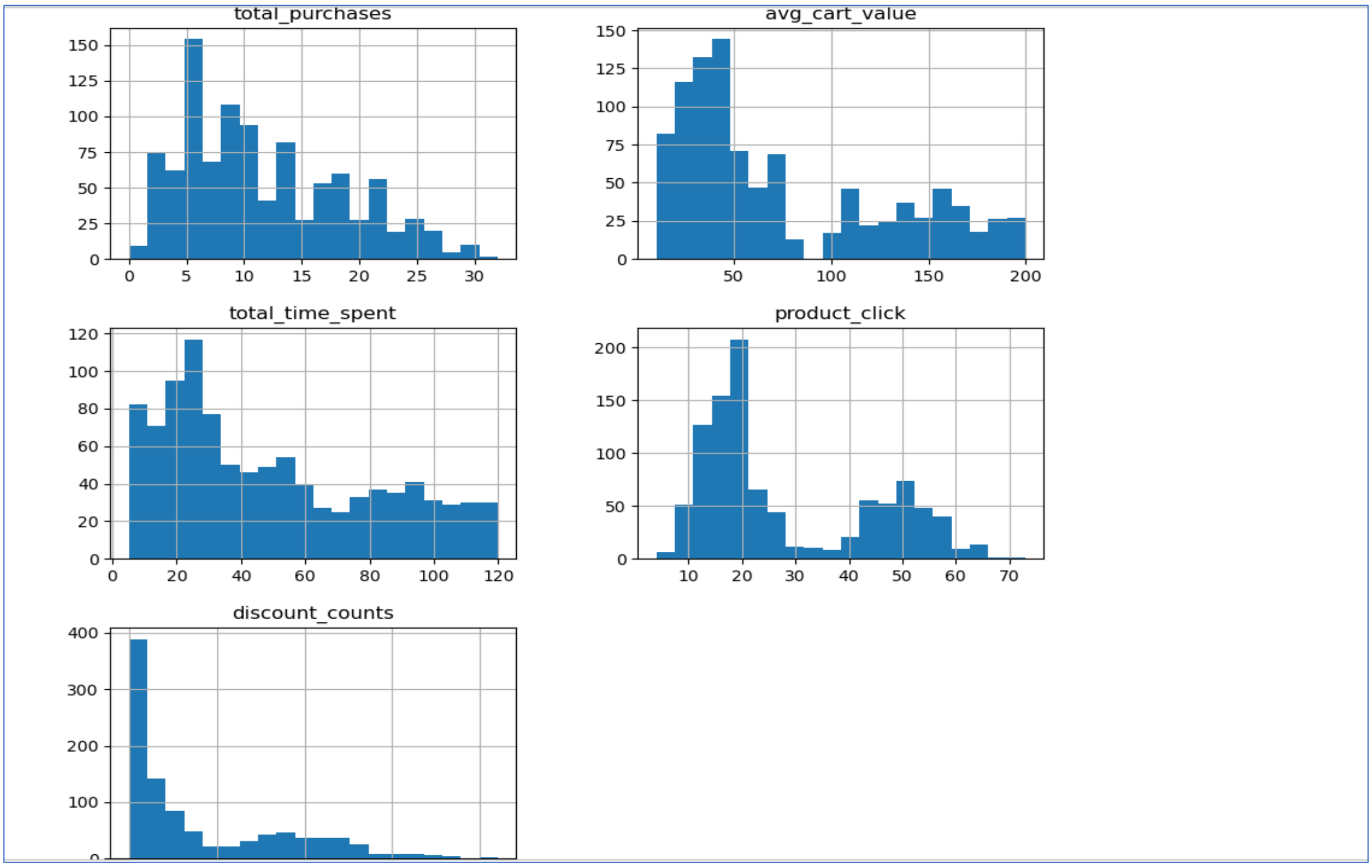


```
[62]: fig, axes = plt.subplots(1, 2, figsize=(12, 5))

      sns.boxplot(x=df['total_purchases'], ax=axes[0])
      axes[0].set_title('Boxplot of Total Purchases')

      sns.boxplot(x=df['avg_cart_value'], ax=axes[1])
      axes[1].set_title('Boxplot of Average Cart Value')
      plt.tight_layout()
      plt.show()
```

Pair plots



Section 03 – Preprocessing & Feature Scaling

3.1 Standardization

Clustering algorithms like K-Means are highly sensitive to the scale of the features, meaning that variables with larger ranges can disproportionately influence the clustering results. To ensure that all features contribute equally, ‘StandardScaler’ from the ‘sklearn.preprocessing’ module was applied to normalize numerical columns. This transformation standardizes the data so that each feature has a mean of 0 and a standard deviation of 1, ensuring uniformity in the clustering process.

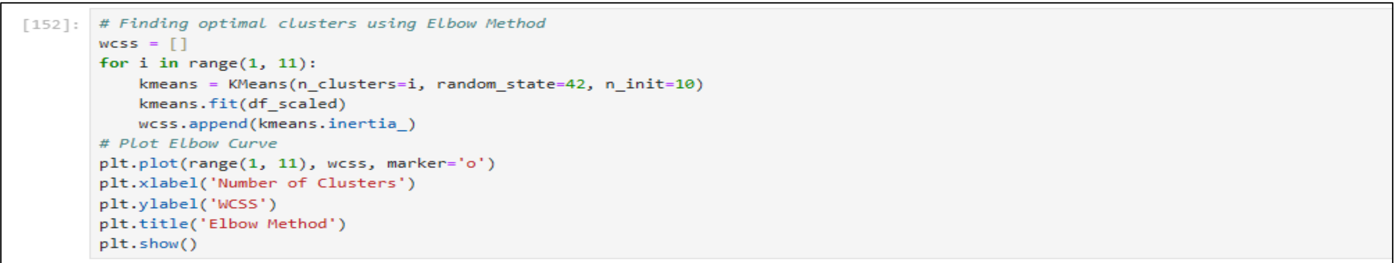
3.2 Justification of Model Choice

After analyzing the dataset, K-Means Clustering was chosen for the following reasons

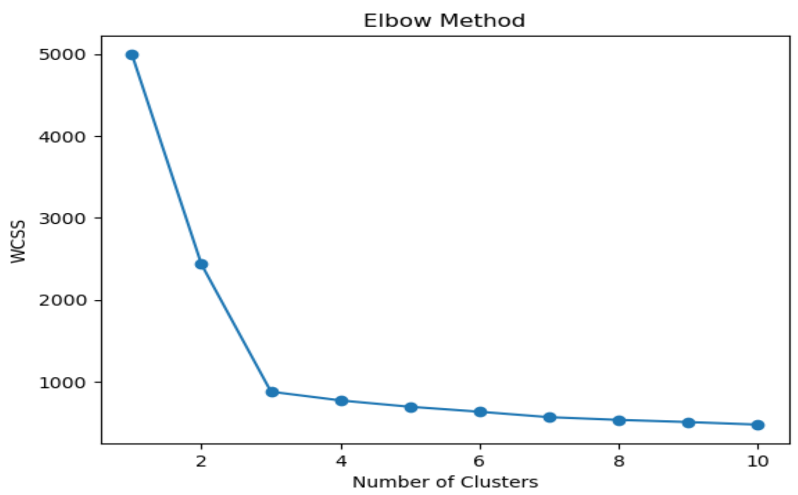
- Scalability - K-Means performs well on moderate to large datasets.
- Efficiency - The algorithm has a linear time complexity of, where is the number of data points and is the number of clusters.
- Interpretability - The output clusters are intuitive and can be easily visualized.
- Assumption of Spherical Clusters - K-Means assumes clusters to be roughly spherical, which aligns well with the distribution of customer behavior.

3.3 Determining Optimal Number of Clusters

The Elbow Method was used to determine the optimal number of clusters for the K-Means model. The Elbow Method was used to determine the optimal number of clusters. A clear bend at k=3 confirmed that three clusters were the best fit.



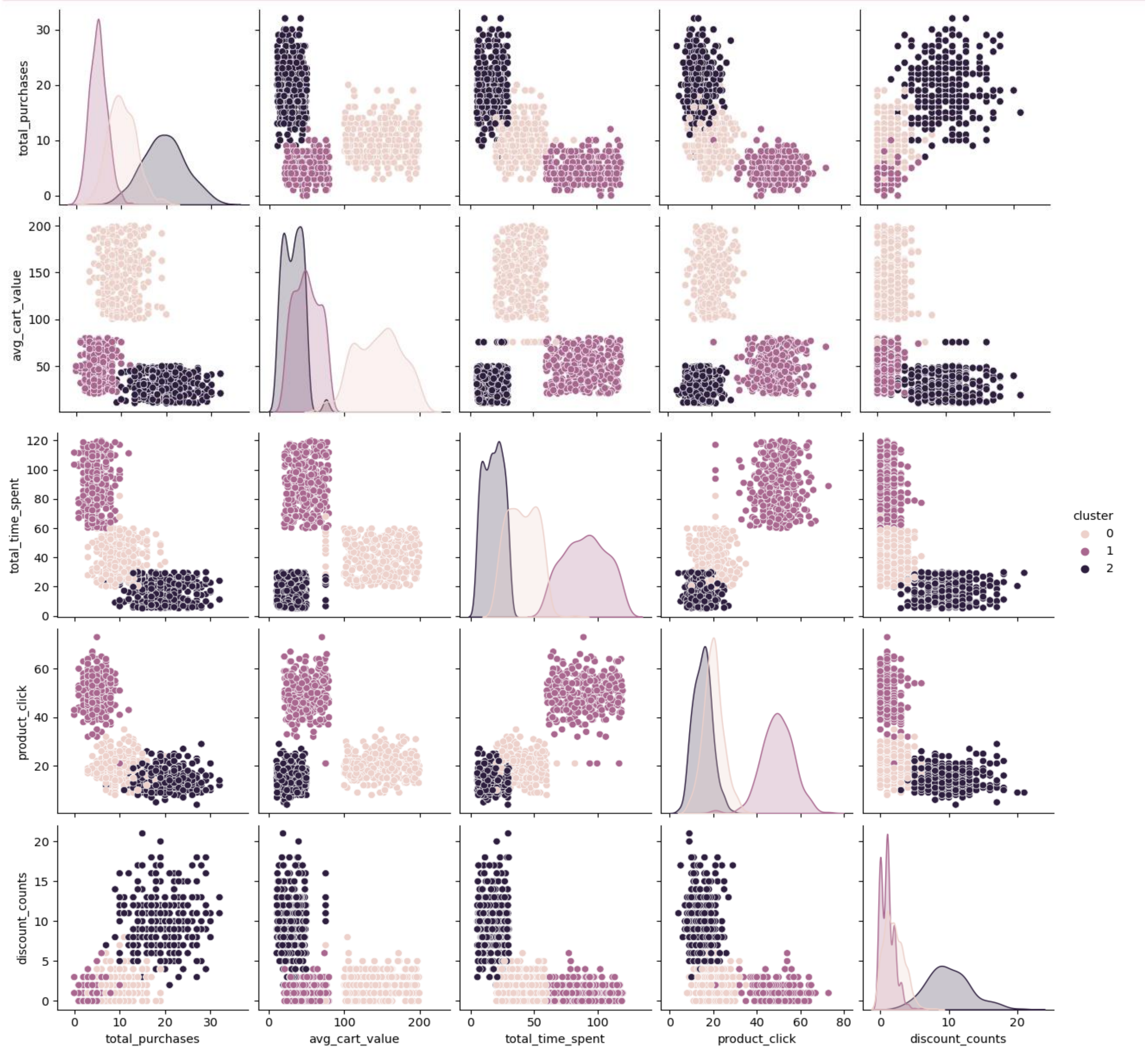
```
[152]:  # Finding optimal clusters using Elbow Method
        wcss = []
        for i in range(1, 11):
            kmeans = KMeans(n_clusters=i, random_state=42, n_init=10)
            kmeans.fit(df_scaled)
            wcss.append(kmeans.inertia_)
        # Plot Elbow Curve
        plt.plot(range(1, 11), wcss, marker='o')
        plt.xlabel('Number of Clusters')
        plt.ylabel('WCSS')
        plt.title('Elbow Method')
        plt.show()
```



3.4 Assigning Customers to Clusters

K-Means clustering successfully identified three groups of customers

- Cluster 0 - Orange (High Spenders) - Moderate purchases, high cart value, low discount usage.
- Cluster 1 - Purple (Bargain Hunters) - High purchases, low cart value, frequent discount usage.
- Cluster 2 - Black (Window Shoppers) - Low purchases, moderate cart value, high browsing time.



Analysis of Cluster Characteristics

1. Total Purchases vs. Other Features

Total Purchases vs. Average Cart Value

- Cluster 0 (High Spenders - Orange) - Forms a distinct group with low-to-moderate total purchases and clearly higher average cart value.
- Cluster 1 (Bargain Hunters - Purple) - Shows a concentration of points with moderate-to-high total purchases but lower average cart values. This is consistent with bargain-hunting behavior.

- Cluster 2 (Window Shoppers - Black) - Shows a mix of total purchases from low end, and moderate range of average cart value.

Total Purchases vs. Total Time Spent

- Cluster 0 (High Spenders - Orange) - low-to-moderate range of total purchases and relative time spent.
- Cluster 1 (Bargain Hunters - Purple) - moderate-to-high range of total purchases and with varying of total time spent.
- Cluster 2 (Window Shoppers - Black) - low range of total purchases with a high total time spent.

Total Purchases vs. Product Clicks

- Cluster 0 (High Spenders - Orange) - low-to-moderate range of total purchases and high amount of product clicks.
- Cluster 1 (Bargain Hunters - Purple) - moderate-to-high range of total purchases and with moderate amounts of product clicks.
- Cluster 2 (Window Shoppers - Black) - low range of total purchases with low amount of product clicks.

Total Purchases vs. Discount Counts

- Cluster 0 (High Spenders - Orange) - low-to-moderate range of total purchases and low discount counts.
- Cluster 1 (Bargain Hunters - Purple) - moderate-to-high range of total purchases and high discount counts.
- Cluster 2 (Window Shoppers - Black) - low range of total purchases with high discount counts.

2. Average Cart Value vs. Other Features

Average Cart Value vs. Total Time Spent

- Cluster 0 (High Spenders - Orange) - high average cart value with a varied range of total time spent.
- Cluster 1 (Bargain Hunters - Purple) - low average cart value with a varied range of total time spent.
- Cluster 2 (Window Shoppers - Black) - with a varied range of average cart value with high amount. of time spent

Average Cart Value vs. Product Clicks

- Cluster 0 (High Spenders - Orange) - high average cart value with high amounts of product clicks.
- Cluster 1 (Bargain Hunters - Purple) - low average cart value with a varied range of product clicks.
- Cluster 2 (Window Shoppers - Black) - with a varied range of average cart value with low amounts of product clicks.

Average Cart Value vs. Discount Counts

- Cluster 0 (High Spenders - Orange) - high average cart value and low discount counts.
- Cluster 1 (Bargain Hunters - Purple) - low average cart value and high discount counts.
- Cluster 2 (Window Shoppers - Black) - varies amount, average cart value and low amounts of discount counts.

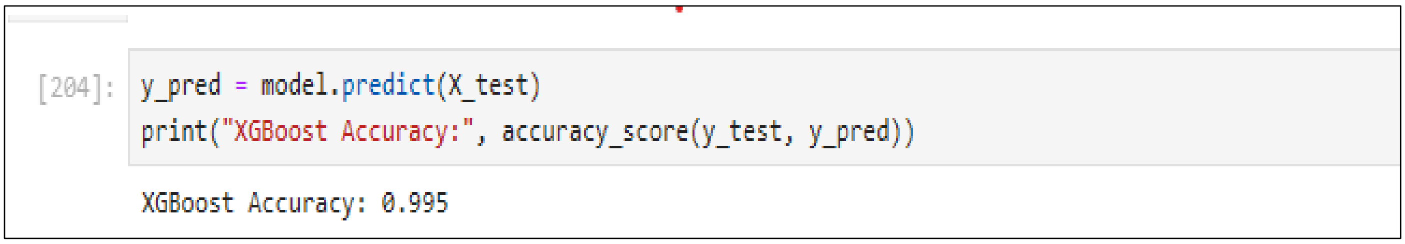
Section 04 - Classification with XGBoost

4.1 Training the Model

After clustering customers into segments using K-Means, we validated the segmentation results using XGBoost, a powerful gradient-boosting classifier. XGBoost is well-suited for structured data and is known for its high performance and efficiency.

4.2 Model Evaluation

The trained XGBoost model was evaluated on the test set, and it achieved an impressive accuracy of **99.5%.** The high accuracy indicates that K-Means clustering effectively separated the customers into well-defined segments, making classification straightforward.



```
[204]: y_pred = model.predict(X_test)
       print("XGBoost Accuracy:", accuracy_score(y_test, y_pred))

       XGBoost Accuracy: 0.995
```

Section 05 - Conclusion

The customer segmentation project successfully identified three meaningful customer groups using K-Means clustering.

The segmentation process revealed the following key insights

- Bargain Hunters frequently purchase low-cost items and actively seek discounts.
- High Spenders make fewer but high-value purchases, prioritizing premium products over discounts.
- Window Shoppers spend significant time browsing but make very few purchases.

To validate the segmentation, an XGBoost classifier was trained, achieving an outstanding **99.5%** accuracy, confirming that the clusters were well-defined.