## **1. Introduction**

I applied K-means clustering to identify patterns in customer behavior, ultimately aiding in targeted marketing strategies and business decision-making.

## **2. Data Preprocessing**

Before clustering, I took the following steps:

* **Concatenate Both Datasets**: The excel file had datasets for both 2009-1020 and 2010-2011. So, I used `pd.concat` to join both into one dataframe.
* **Handling Missing Values**: Approximately 107,927 Customer ID values were missing, and these rows were removed.

| data1 = pd.read\_csv("2009\_2010\_retail.csv") data1 = data1.dropna() data2 = pd.read\_csv("2010\_2011\_retail.csv") dataframes = [data1, data2] data3 = pd.concat(dataframes) print(data3.shape) |
| --- |

* **Feature Engineering**: I derived key features such as:
  + **Total Spending** = Sum of all purchases per customer.
  + **Total Orders** = Count of unique invoices per customer.(number of orders they’ve made)
  + **Total Quantity** = Sum of all items purchased per customer.
  + **Average Order Value** = Total Spending / Total Orders.(Amount spent per order made)
  + **Average Items Per Order** = Total Quantity / Total Orders. (Average items per order)
  + **Distinct Items Purchased** = Number of unique products bought by a customer.
  + **Recency** = Days since the last purchase.

| data3['Customer ID'] = data3['Customer ID'].astype(str) data3['InvoiceData'] = pd.to\_datetime(data3['InvoiceDate']) data3 = data3[data3['Quantity'] >0 ] data = data3.groupby('Customer ID').agg(  Total\_Spending=('Price', lambda x: (x \* data3.loc[x.index, 'Quantity']).sum()),  Total\_Orders=('Invoice', 'nunique'),  Total\_Quantity=('Quantity', 'sum'),  Distinct\_Items=('StockCode', 'nunique'),  First\_Purchase=('InvoiceDate', 'min'),  Last\_Purchase=('InvoiceDate', 'max') ).reset\_index()  latest\_purchase = data3['InvoiceDate'].max()    data['Last\_Purchase'] = pd.to\_datetime(data['Last\_Purchase'], errors='coerce') data['First\_Purchase'] = pd.to\_datetime(data['First\_Purchase'], errors='coerce') latest\_purchase = pd.to\_datetime(latest\_purchase, errors='coerce')  data['Recency'] = (latest\_purchase - data['Last\_Purchase']).dt.days data.drop(columns=['First\_Purchase', 'Last\_Purchase'], inplace=True) data['Avg\_Order\_Value'] = data['Total\_Spending'] / data['Total\_Orders'] data['Avg\_Items\_Per\_Order'] = data['Total\_Quantity'] / data['Total\_Orders'] |
| --- |

* **Standardization**: Since the dataset contained numerical variables on different scales, I applied **StandardScaler, Log Values and Winsorization**  to normalize the data. Winsorization was used to remove outliers in the 80th percentile.



| data\_log = data.copy() data\_log['Total\_Spending'] = np.log1p(data\_log['Total\_Spending']) data\_log['Total\_Quantity'] = np.log1p(data\_log['Total\_Quantity']) data\_log['Distinct\_Items'] = np.log1p(data\_log['Distinct\_Items']) data\_log['Avg\_Order\_Value'] = np.log1p(data\_log['Avg\_Order\_Value']) data\_log['Avg\_Items\_Per\_Order'] = np.log1p(data\_log['Avg\_Items\_Per\_Order']) data\_log['Total\_Orders'] = np.log1p(data\_log['Total\_Orders']) data\_log['Recency'] = np.log1p(data\_log['Recency']) cols\_to\_winsorize = ['Total\_Spending', 'Total\_Orders', 'Total\_Quantity',   'Distinct\_Items', 'Avg\_Items\_Per\_Order', 'Avg\_Order\_Value']  data\_log['Recency'] = data\_log['Recency'].replace(-np.inf, np.nan)  # Ensure all columns are numeric for col in cols\_to\_winsorize:  data\_log[col] = pd.to\_numeric(data\_log[col], errors='coerce')  # Winsorize: Cap values at the 80th percentile for col in cols\_to\_winsorize:   q8 = data\_log[col].quantile(0.80)  data\_log[col] = np.where(data\_log[col] > q8, q8, data\_log[col])  x = data\_log[['Total\_Spending', 'Total\_Orders', 'Total\_Quantity',   'Avg\_Order\_Value', 'Avg\_Items\_Per\_Order', 'Distinct\_Items', 'Recency']]  from sklearn.preprocessing import StandardScaler  scaler = StandardScaler() X\_scaled = scaler.fit\_transform(x) |
| --- |

## **3. Clustering Methodology**

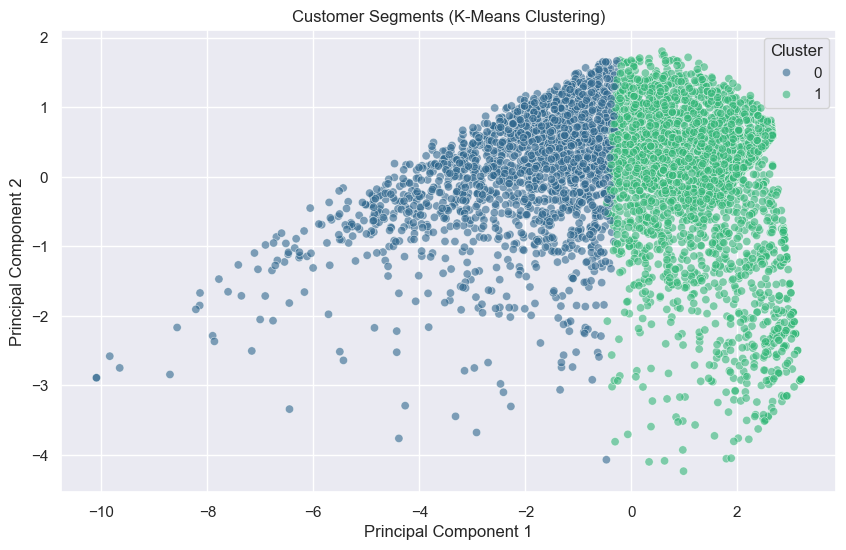
I used **K-Means Clustering** to segment the customers. The process included:

* **Choosing the Optimal K**:
  + The **Elbow Method** was used to determine the best number of clusters.
  + Based on the analysis, I selected **K = 2** as the optimal number of clusters.
* **Applying K-Means**: The algorithm was implemented on the PCA-transformed data, ensuring efficiency and interpretability.

| wcss = []  for i in range(1,10):  kmeans = KMeans(i)  kmeans.fit(X\_scaled)  wcss.append(kmeans.inertia\_)   wcss  plt.plot(range(1,10), wcss) plt.xlabel("number of clusters") plt.ylabel('wcss')   kmeans = KMeans(n\_clusters=2, random\_state=42) data\_log['Cluster'] = kmeans.fit\_predict(X\_scaled)  identified\_clusters = kmeans.fit\_predict(X\_scaled) identified\_clusters  data\_with\_clusters = data\_log.copy() data\_with\_clusters['Cluster'] = identified\_clusters data\_with\_clusters |
| --- |

## **4. Results & Analysis**

| import matplotlib.pyplot as plt import seaborn as sns from sklearn.decomposition import PCA  # Reduce to 2D for visualization pca = PCA(n\_components=2) X\_pca = pca.fit\_transform(X\_scaled)  # Add PCA components to the DataFrame data\_with\_clusters['PCA1'] = X\_pca[:, 0] data\_with\_clusters['PCA2'] = X\_pca[:, 1]  # Scatter plot of clusters plt.figure(figsize=(10, 6)) sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', palette='viridis', data=data\_with\_clusters, alpha=0.6) plt.title('Customer Segments (K-Means Clustering)') plt.xlabel('Principal Component 1') plt.ylabel('Principal Component 2') plt.legend(title="Cluster") plt.show() |
| --- |



| data\_with\_clusters['Cluster'] = identified\_clusters  cluster\_summary = data\_with\_clusters.groupby("Cluster").mean(numeric\_only=True) print(cluster\_summary) |
| --- |

### **Cluster Characteristics**

| **Metric** | **Cluster 0** | **Cluster 1** |
| --- | --- | --- |
| Total Spending | 6.25 | 8.37 |
| Total Orders | 0.93 | 1.74 |
| Total Quantity | 4.78 | 6.77 |
| Distinct Items | 2.63 | 4.26 |
| Recency | 5.57 days | 4.99 days |
| Avg Order Value | 5.84 | 6.75 |
| Avg Items Per Order | 4.39 | 5.19 |

### **Key Insights**

* **Cluster 1** customers tend to purchase more items and spend more per order.
* **Cluster 0** customers exhibit lower spending
* Customers in Cluster 1 made purchases more recently (lower Recency value), indicating more engagement with the store.
* The business can target Cluster 1 customers with loyalty programs, rewards and others to encourage them to keep coming, while Cluster 0 customers may require re-engagement strategies, discounts, sales and targeted ads to make them spend more.

## **4. Conclusion & Business Recommendations**

* **Target High-Value Customers (Cluster 1)**: Implement exclusive promotions and personalized discounts.
* **Re-engage Low-Activity Customers (Cluster 0)**: Utilize email marketing, limited-time offers, and reminders to encourage purchases.
* **Monitor Customer Behavior**: Regularly update clustering models to reflect changing consumer habits.