## **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', 'Country').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()  data['Country'] = data['Country'].astype('category').cat.codes  country\_mapping = dict(enumerate(data3['Country'].astype('category').cat.categories)) print(country\_mapping)   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['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 MinMaxScaler from sklearn.decomposition import PCA    scaler = MinMaxScaler() X\_scaled = scaler.fit\_transform(x[features])  pca = PCA(n\_components=2) x\_pca = pca.fit\_transform(X\_scaled)  X\_scaled\_ = pd.DataFrame(x\_pca, columns=['PCA1', 'PCA2']) |
| --- |

## **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 = 6** as the optimal number of clusters.
* **Applying K-Means**: The algorithm was implemented on the PCA-transformed data, ensuring efficiency and interpretability.

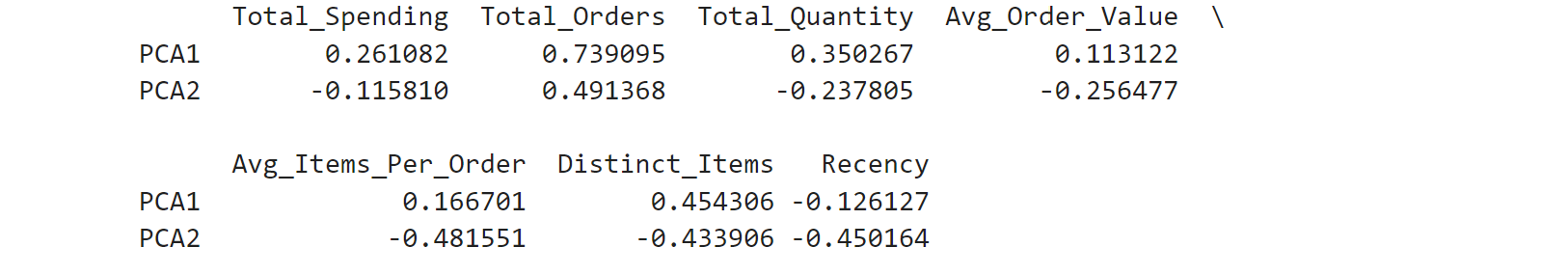
| wcss = []  for i in range(1,10):  kmeans = KMeans(n\_clusters=i, random\_state=42, n\_init=10)  kmeans.fit(X\_scaled\_)  wcss.append(kmeans.inertia\_)   wcss  plt.plot(range(1,10), wcss) plt.xlabel("number of clusters") plt.ylabel('wcss')  kmeans = KMeans(n\_clusters=6, random\_state=42, n\_init=10) identified\_clusters = kmeans.fit\_predict(X\_scaled\_) x['Cluster'] = identified\_clusters X\_scaled\_['Cluster'] = identified\_clusters |
| --- |

## **4. Results & Analysis**

| import matplotlib.pyplot as plt import seaborn as sns import matplotlib.pyplot as plt import seaborn as sns  plt.figure(figsize=(10, 6)) sns.scatterplot(  x='PCA1', y='PCA2', hue='Cluster', palette='viridis',   data=X\_scaled\_, 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() |
| --- |



| pca\_components = pd.DataFrame(pca.components\_, columns=features, index=['PCA1', 'PCA2']) print(pca\_components) |
| --- |



| cluster\_summary = x.groupby("Cluster")[features].mean() print(cluster\_summary) |
| --- |

### 

### 

### **Key Insights:**

##### **Cluster 0: "Frequent Big Spenders"**

##### High Spending, High Orders, High Variety

##### Low Recency (very recent customers)

##### **Cluster 1: "Occasional Buyers"**

##### Average Spendings, Low Orders, Low Variety(Low Distinct Items)

##### High Recency (older customers)

##### **Cluster 2: "Balanced or Regular Buyers"**

##### Average Spendings, Average Orders, Average Variety(Average Distinct Items)

##### Low Recency (Recentcustomers)

##### **Cluster 3: "Loyal but Rare Buyers"**

##### Average Spendings, Low Orders, Average Variety(Average Distinct Items)

##### High Recency (Old customers)

##### spend well when they buy

##### **Cluster 4: "Inactive Customers"**

##### Low Spending, Low Orders, Very Low Variety

##### Highest Recency (Oldest customers)

##### **Cluster 5: "High-Value Customers"**

##### High Spending, Average Orders, High Variety

##### High Recency (not very recent customers)

##### Spend a lot, buy different items, but not as often as Cluster 0.

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

* **Target High-Value Customers (Cluster 0 and 5)**: Implement exclusive promotions and personalized discounts.
* **Re-engage Low-Activity Customers (Cluster 1 & 3)**: Utilize email marketing, limited-time offers, and reminders to encourage purchases.
* **Reactivate Inactive Customers(Cluster 4)**: free shipping offers, or surveys to understand why they left.
* **Nurture Average Customers (Cluster 2)**: Offer a subscription model or loyalty rewards to increase their frequency.